EACL-2006

11th Conference
of the European Chapter of the
Association for Computational Linguistics

Proceedings of the workshop on

Learning Structured Information in Natural Language Applications

April, 3, 2006
Trento, Italy

SPONSOR:

Department of Computer Science, University of Rome "Tor Vergata"

ORGANIZING COMMITTEE:

Roberto Basili, University of Rome "Tor Vergata", Co-chair
Alessandro Moschitti, University of Rome "Tor Vergata", Co-chair

PROGRAM COMMITTEE:

Nicola Cancedda (Xerox Research Centre Europe, France)
Nello Cristianini (University of California, Davis , USA)
Aron Culotta (University of Massachusetts Amherst, USA)
Walter Daelemans (University of Antwerp, Netherlands)
Marcello Federico (ITC-Irst, Italy)
Attilio Giordana (University of Turin, Italy)
Marko Grobelink (J. Stefan Institute, Ljubljana, Slovenia)
Fred Jelinek (CLSP John Hopkins University, USA)
Thorsten Joachims (Cornell University, USA)
Lluis Marquez (Universitat Politecnica de Catalunya, Spain)
Giuseppe Riccardi (University of Trento, Italy)
Dan Roth (University of Illinois at Urbana-Champaign, USA)
Alex Smola (National ICT Australia, ANU)
Carlo Strapparava (ITC-Irst, Italy)
John Shawe Taylor (University of Southampton, UK)
Ben Taskar (University of California at Berkeley , USA)
Dimitry Zelenko (SRA international inc., USA)

WORKSHOP WEBSITE:

`http://ai-nlp.info.uniroma2.it/eacl2006-ws10/`

INTRODUCTION

Language processing largely deals with multidimensional and highly structured forms of information. Indeed, from the morphological up to the deep syntactic and semantic levels, linguistic information is often described by structured data, making the learning of the associated linguistic tasks more complex.

Traditional methods for the design of language applications involve the extraction of features that map data representations to vectors of attributes/values. Unfortunately, there is no methodology that helps in this feature modeling problem. Consequently, in order to encode structured data, the designer has to rely on his/her deep knowledge, expertise and intuition about the linguistic phenomenon associated with the target structures.

Recently, approaches that attempt to alleviate such modeling complexity by directly encoding structured data have been developed. Among other, kernel methods and conditional random fields provide interesting properties. The former use kernel functions to implicitly define richer feature spaces (e.g. substructure spaces) whereas the latter allow the designer to directly encode the probabilistic model on the structures. The promising aspects of such approaches open new research directions:

(a) the study of their impact on the modeling of diverse natural language structures,

(b) their comparative assessment with traditional attribute-value models, and

(c) the investigation of techniques which aim to improve their efficiency.

Additionally, the complementary study of mapping the classification function in structured spaces is very interesting. Classification functions can be designed to output structured data instead of simple values. In other words, the output values may be interpreted as macro-labels which describe configurations and dependencies over simpler components, e.g. parse trees or semantic structures.

The workshop was held on April 3, 2006, just preceding the 11th Conference of the European Chapter of the Association for Computational Linguistics. Its primary objective was to favor the discussing on the above topics. For this purpose, researchers from different communities such as machine learning, computational linguistics, information retrieval and data mining were invited to partecipate at the workshop to promote the discussion and development of new ideas and methods for the effective exploitation of "structured data" for natural language learning and applications.

Regarding these latter, *Coreference Resolution*, *Information/Relation Extraction*, *Machine Translation*, *Multilingual Corpus Alignment*, *Named Entity Recognition*, *Question Classification*, *Semantic Role Labeling*, *Semantic Parsing*, *Syntactic Parsing and Parse Re-Ranking*, *Text Categorization* and *Word Sense Disambiguation* were considered particularly interesting for the workshop discussion. Moreover, machine learning approaches based on *Kernel Methods*, *Maximal Margin Classifiers*, *Conditional Random Fields* and *Support Vector Machines* (SVMs) were judged those most promising to deal with structured data.

This volume contains twelve papers accepted for presentation at the workshop. We received a rather large number of high quality papers. Consequently, due to the restriction imposed by one day workshop, we decided to divide the papers in two categories: those reporting almost conclusive results and/or theories supported by a sound experimentation (full papers) and those proposing preliminary results and/or theories that would have been received significant benefits from a more extensive experimentation (position papers).

The program committee accepted eight submissions as full papers (about 50% of acceptance rate) and others four as position papers. The workshop papers deal with several interesting aspects of structured data in natural language learning. From a machine learning perspective, the contributions on: kernel methods within SVMs, probabilistic approaches and unsupervised methods, e.g. latent semantic analysis, support an interesting comparative discussion. Regarding the NLP tasks, the papers touch almost all the targeted applications: *Named Entity recognition*, *Relation Extraction*, *Discourse Parsing*, *Semantic Role Labeling*, *Prepositional Phrase Attachment problem*, *Text Categorization*, *Machine Translation* and *Question Answering*.

We believe that the workshop outcome will be helpful to increase the knowledge about advanced machine learning techniques in the modeling of structured data for Natural Language Applications.

We gratefully acknowledge all the members of the Program Committee for the excellent work done in reviewing and commenting the individual submissions within the very short time.

Roberto Basili
Alessandro Moschitti

Rome, February 15th, 2006.

WORKSHOP PROGRAM

Monday, April 3

9:00-9:15	WELCOME AND INTRODUCTORY NOTES
	FULL PAPER SESSION
9:15-9:40	*Maximum Entropy Tagging with Binary and Real-Valued Features* Vanessa Sandrini, Marcello Federico and Mauro Cettolo
9:40-10:05	*Constraint Satisfaction Inference:* *Non-probabilistic Global Inference for Sequence Labelling* Sander Canisius, Antal van den Bosch and Walter Daelemans
10:05-10:30	*Decomposition Kernels for Natural Language Processing* Fabrizio Costa, Sauro Menchetti, Alessio Ceroni, Andrea Passerini and Paolo Frasconi
10:30-11:00	COFFEE BREAK
11:05-11:50	*Invited Speaker*
	POSITION PAPER SESSION
11:50-12:10	*A Multiclassifier based Document Categorization System:* *profiting from the Singular Value Decomposition Dimensionality Reduction Technique* Ana Zelaia, Iñaki Alegria, Olatz Arregi and Basilio Sierra
12:10-12:30	*Discourse Parsing: Learning FOL Rules based on Rich Verb Semantic Representations to automatically label Rhetorical Relations* Rajen Subba, Barbara Di Eugenio and Su Nam Kim
12:30-14:00	LAUNCH BREAK
	FULL PAPER SESSION
14:00-14:25	*Reranking Translation Hypotheses Using Structural Properties* Saša Hasan, Oliver Bender, and Hermann Ney
14:25-14:50	*Tree Kernel Engineering in Semantic Role Labeling Systems* Alessandro Moschitti, Daniele Pighin and Roberto Basili
14:50-15:15	*Syntagmatic Kernels: a Word Sense Disambiguation Case Study* Claudio Giuliano, Alfio Gliozzo and Carlo Strapparava

	POSITION PAPER SESSION
15:15-15:35	*Learning to Identify Definitions using Syntactic Features* Ismail Fahmi and Gosse Bouma
15:35-15:55	*An Ontology-Based Approach to Disambiguation of Semantic Relations* Tine Lassen and Thomas Vestskov Terney
15:55-16:30	COFFEE BREAK
	FULL PAPER SESSION
16:30-16:55	*Towards Free-text Semantic Parsing:* *A Unified Framework Based on FrameNet, VerbNet and PropBank* Ana-Maria Giuglea and Alessandro Moschitti
16:55-17:20	*Constructing a Rule Based Naming System for Thai Names Using the Concept of Ontologies* Chakkrit Snae
17:20-18:00	CONCLUSIVE REMARKS AND DISCUSSION

Table of Contents

Maximum Entropy Tagging with Binary and Real-Valued Features

Vanessa Sandrini **Marcello Federico** **Mauro Cettolo**
ITC-irst - Centro per la Ricerca Scientifica e Tecnologica
38050 Povo (Trento) - ITALY
{surname}@itc.it

Abstract

Recent literature on text-tagging reported successful results by applying Maximum Entropy (ME) models. In general, ME taggers rely on carefully selected binary features, which try to capture discriminant information from the training data. This paper introduces a standard setting of binary features, inspired by the literature on named-entity recognition and text chunking, and derives corresponding real-valued features based on smoothed log-probabilities. The resulting ME models have orders of magnitude fewer parameters. Effective use of training data to estimate features and parameters is achieved by integrating a leaving-one-out method into the standard ME training algorithm. Experimental results on two tagging tasks show statistically significant performance gains after augmenting standard binary-feature models with real-valued features.

1 Introduction

The Maximum Entropy (ME) statistical framework (Darroch and Ratcliff, 1972; Berger et al., 1996) has been successfully deployed in several NLP tasks. In recent evaluation campaigns, e.g. DARPA IE and CoNLL 2000-2003, ME models reached state-of-the-art performance on a range of text-tagging tasks.

With few exceptions, best ME taggers rely on carefully designed sets of features. Features correspond to binary functions, which model events, observed in the (annotated) training data and supposed to be meaningful or discriminative for the task at hand. Hence, ME models result in a log-linear combination of a large set of features, whose weights can be estimated by the well known Generalized Iterative Scaling (GIS) algorithm by Darroch and Ratcliff (1972).

Despite ME theory and its related training algorithm (Darroch and Ratcliff, 1972) do not set restrictions on the *range* of feature functions[1], popular NLP text books (Manning and Schutze, 1999) and research papers (Berger et al., 1996) seem to limit them to binary features. In fact, only recently, log-probability features have been deployed in ME models for statistical machine translation (Och and Ney, 2002).

This paper focuses on ME models for two text-tagging tasks: Named Entity Recognition (NER) and Text Chuncking (TC). By taking inspiration from the literature (Bender et al., 2003; Borthwick, 1999; Koeling, 2000), a set of *standard* binary features is introduced. Hence, for each feature type, a corresponding real-valued feature is developed in terms of smoothed probability distributions estimated on the training data. A direct comparison of ME models based on binary, real-valued, and mixed features is presented. Besides, performance on the tagging tasks, complexity and training time by each model are reported. ME estimation with real-valued features is accomplished by combining GIS with the leave-one-out method (Manning and Schutze, 1999).

Experiments were conducted on two publicly available benchmarks for which performance levels of many systems are published on the Web. Results show that better ME models for NER and TC can be developed by integrating binary and real-valued features.

[1] Darroch and Ratcliff (1972) show how any set of real-valued feature functions can be properly handled.

2 ME Models for Text Tagging

Given a sequence of words $w_1^T = w_1, \ldots, w_T$ and a set of tags $\mathcal{C}$, the goal of text-tagging is to find a sequence of tags $c_1^T = c_1, \ldots, c_T$ which maximizes the posterior probability, i.e.:

$$\hat{c_1^T} = \arg\max_{c_1^T} p(c_1^T \mid w_1^T). \quad (1)$$

By assuming a discriminative model, Eq. (1) can be rewritten as follows:

$$\hat{c_1^T} = \arg\max_{c_1^T} \prod_{t=1}^{T} p(c_t \mid c_1^{t-1}, w_1^T), \quad (2)$$

where $p(c_t|c_1^{t-1}, w_1^T)$ is the target conditional probability of tag c_t given the context (c_1^{t-1}, w_1^T), i.e. the entire sequence of words and the full sequence of previous tags. Typically, independence assumptions are introduced in order to reduce the context size. While this introduces some approximations in the probability distribution, it considerably reduces data sparseness in the sampling space. For this reason, the context is limited here to the two previous tags (c_{t-2}^{t-1}) and to four words around the current word (w_{t-2}^{t+2}). Moreover, limiting the context to the two previous tags permits to apply dynamic programming (Bender et al., 2003) to efficiently solve the maximization (2).

Let $y = c_t$ denote the class to be guessed ($y \in \mathcal{Y}$) at time t and $x = c_{t-2}^{t-1}, w_{t-2}^{t+2}$ its context ($x \in \mathcal{X}$). The generic ME model results:

$$p_\lambda(y \mid x) = \frac{\exp(\sum_{i=1}^{n} \lambda_i f_i(x, y))}{\sum_{y'} \exp(\sum_{i=1}^{n} \lambda_i f_i(x, y'))}. \quad (3)$$

The n feature functions $f_i(x, y)$ represent any kind of information about the event (x, y) which can be useful for the classification task. Typically, binary features are employed which model the verification of simple events within the target class and the context.

In Mikheev (1998), binary features for text tagging are classified into two broad classes: *atomic* and *complex*. Atomic features tell information about the current tag and one single item (word or tag) of the context. Complex features result as a combination of two or more atomic features. In this way, if the grouped events are not independent, complex features should capture higher correlations or dependencies, possibly useful to discriminate.

In the following, a standard set of binary features is presented, which is generally employed for text-tagging tasks. The reader familiar with the topic can directly check this set in Table 1.

3 Standard Binary Features

Binary features are indicator functions of specified events of the sample space $\mathcal{X} \times \mathcal{Y}$. Hence, they take value 1 if the event occurs or 0 otherwise. For the sake of notation, the feature name denotes the type of event, while the index specifies its parameters. For example:

$$\texttt{Orth}_{\texttt{person,Cap},-1}(x, y)$$

corresponds to an *Orthographic* feature which is active if and only if the class at time t is `person` and the word at time $t-1$ in the context starts with capitalized letter.

3.1 Atomic Features

Lexical features These features model co-occurrences of classes and single words of the context. Lexical features are defined on a window of ± 2 positions around the current word. Lexical features are denoted by the name `Lex` and indexed with the triple c, w, d which fixes the current class, i.e. $c_t = c$, the identity and offset of the word in the context, i.e. $w_{t+d} = w$. Formally, the feature is computed by:

$$\texttt{Lex}_{c,w,d}(x, y) \hat{=} \delta(c_t = c) \cdot \delta(w_{t+d} = w).$$

For example, the lexical feature for word `Verona`, at position t with tag `loc` (location) is:

$$\texttt{Lex}_{\texttt{loc,Verona},0}(x, y) = \delta(c_t = \texttt{loc}) \cdot \cdot\delta(w_t = \texttt{Verona}).$$

Lexical features might introduce data sparseness in the model, given that in real texts an important fraction of words occur only once. In other words, many words in the test set will have no corresponding features-parameter pairs estimated on the training data. To cope with this problem, all words observed only once in the training data were mapped into the special symbol `oov`.

Syntactic features They model co-occurrences of the current class with part-of-speech or chunk tags of a specific position in the context. Syntactic features are denoted by the name `Syn` and indexed with a 4-tuple $(c, \texttt{Pos}, p, d)$ or $(c, \texttt{Chnk}, p, d)$,

Name	Index	Definition
Lex	c, w, d	$\delta(c_t = c) \cdot \delta(w_{t+d} = w), d \in Z$
Syn	c, T, p, d	$\delta(c_t = c) \cdot \delta(\mathtt{T}(w_{t+d}) = p)$, $\mathtt{T} \in \{\mathtt{Pos}, \mathtt{Chnk}\}, d \in Z$
Orth	c, F, d	$\delta(c_t = c) \cdot \mathtt{F}(w_{t+d})$, $\mathtt{F} \in \{\mathtt{IsCap}, \mathtt{IsCAP}\}, d \in Z$
Dict	c, L, d	$\delta(c_t = c) \cdot \mathtt{InList}(L, w_{t+d}), d \in Z$
Tran	c, c', d	$\delta(c_t = c) \cdot \delta(c_{t-d} = c')$ $\quad d \in N^+$
Lex+	c, s, k, w_s^{s+k-1}	$\prod_{d=s}^{s+k-1} \mathtt{Lex}_{c,w_d,d}(x, y), k \in N^+, s \in Z$
Syn+	c, T, s, k, p_s^{s+k-1}	$\prod_{d=s}^{s+k-1} \mathtt{Syn}_{c,T,p_d,d}(x, y), k \in N^+, s \in Z$
Orth+	c, F, k, b_{-k}^{+k}	$\delta(c_t = c) \cdot \prod_{d=-k}^{k} \delta(\mathtt{Orth}_{c,F,d}(x, y) = b_d)$, $b_d \in \{0, 1\}, k \in N^+$
Dict+	c, L, k, b_{-k}^{+k}	$\delta(c_t = c) \cdot \prod_{d=-k}^{k} \delta(\mathtt{Dict}_{c,L,d}(x, y) = b_d)$, $b_d \in \{0, 1\}, k \in N^+$
Tran+	c, k, c_1^k	$\prod_{d=1'}^{k} \mathtt{Tran}_{c,c_d,d}(x, y)$ $\quad k \in N^+$

Table 1: Standard set of binary features for text tagging.

which fixes the class c_t, the considered syntactic information, and the tag and offset within the context. Formally, these features are computed by:

$$\mathtt{Syn}_{c,\mathtt{Pos},p,d}(x, y) \hat{=} \delta(c_t = c) \cdot \delta(\mathtt{Pos}(w_{t+d}) = p)$$

$$\mathtt{Syn}_{c,\mathtt{Chnk},p,d}(x, y) \hat{=} \delta(c_t = c) \cdot \delta(\mathtt{Chnk}(w_{t+d}) = p).$$

Orthographic features These features model co-occurrences of the current class with surface characteristics of words of the context, e.g. check if a specific word in the context starts with capitalized letter (IsCap) or is fully capitalized (IsCAP). In this framework, only capitalization information is considered. Analogously to syntactic features, orthographic features are defined as follows:

$$\mathtt{Orth}_{c,\mathtt{IsCap},d}(x, y) \hat{=} \delta(c_t = c) \cdot \mathtt{IsCap}(w_{t+d})$$

$$\mathtt{Orth}_{c,\mathtt{IsCAP},d}(x, y) \hat{=} \delta(c_t = c) \cdot \mathtt{IsCAP}(w_{t+d}).$$

Dictionary features These features check if specific positions in the context contain words occurring in some prepared list. This type of feature results relevant for tasks such as NER, in which *gazetteers* of proper names can be used to improve coverage of the training data. Atomic dictionary features are defined as follows:

$$\mathtt{Dict}_{c,L,d}(x, y) \hat{=} \delta(c_t = c) \cdot \mathtt{InList}(L, w_{t+d})$$

where L is a specific pre-compiled list, and InList is a function which returns 1 if the specified word matches one of the multi-word entries of list L, and 0 otherwise.

Transition features Transition features model Markov dependencies between the current tag and a previous tag. They are defined as follows:

$$\mathtt{Tran}_{c,c',d}(x, y) \hat{=} \delta(c_t = c) \cdot \delta(c_{t-d} = c').$$

3.2 Complex Features

More complex events are defined by combining two or more atomic features in one of two ways. $Product$ features take the intersection of the corresponding atomic events. $Vector$ features consider all possible outcomes of the component features.

For instance, the product of 3 atomic Lexical features, with class c, offsets $-2, -1, 0$, and words v_{-2}, v_{-1}, v_0, is:

$$\mathtt{Lex}+_{c,-2,3,v_{-2},v_{-1},v_0}(x, y) \hat{=} \prod_{d=-2}^{0} \mathtt{Lex}_{c,v_d,d}(x, y).$$

Vector features obtained from three Dictionary features with the same class c, list L, and offsets, respectively, -1,0,+1, are indexed over all possible binary outcomes b_{-1}, b_0, b_1 of the single atomic features, i.e.:

$$\mathtt{Dict}+_{c,L,1,b_{-1},b_0,b_{+1}}(x, y) \hat{=} \delta(c_t = c) \times \prod_{d=-1}^{1} \delta(\mathtt{Dict}_{c,L,d}(x, y) = b_d).$$

Complex features used in the experiments are described in Table 1.

The use of complex features significantly increases the model complexity. Assuming that there are $10,000$ words occurring more than once in the training corpus, the above lexical feature potentially adds $O(|\mathcal{C}|10^{12})$ parameters!

As complex binary features might result prohibitive from a computational point of view, real-valued features should be considered as an alternative.

Feature	Index	Probability Distribution
Lex	d	$p(c_t \mid w_{t+d})$
Syn	T, d	$p(c_t \mid \mathtt{T}(w_{t+d}))$
Orth	F, d	$p(c_t \mid \mathtt{F}(w_{t+d}))$
Dict	$List, d$	$p(c_t \mid \mathtt{IsIn}(List, w_{t+d}))$
Tran	d	$p(c_t \mid c_{t-d})$
Lex+	s, k	$p(c_t \mid w_{t+s}, .., w_{t+s+k-1}$
Syn+	T, s, k	$p(c_t \mid \mathtt{T}(w_{t+s}, \ldots, w_{t+s+k-1}))$
Orth+	k, F	$p(c_t \mid \mathtt{F}(w_{t-k}), \ldots, \mathtt{F}(w_{t+k}))$
Dict+	k, L	$p(c_t \mid \mathtt{InList}(\mathtt{L}, \mathtt{w_{t-k}}), \ldots, \mathtt{InList}(\mathtt{L}, \mathtt{w_{t+k}}))$
Tran+	k	$p(c_t \mid c_{t-k}, \ldots, c_{t+k}))$

Table 2: Corresponding standard set of real-values features.

4 Real-valued Features

A binary feature can be seen as a probability measure with support set made of a single event. According to this point of view, we might easily extend binary features to probability measures defined over larger event spaces. In fact, it results convenient to introduce features which are logarithms of conditional probabilities. It can be shown that in this way linear constraints of the ME model can be interpreted in terms of Kullback-Leibler distances between the target model and the conditional distributions (Klakow, 1998).
Let $p_1(y|x), p_2(y|x), \ldots, p_n(y|x)$ be n different conditional probability distributions estimated on the training corpus. In our framework, each conditional probability p_i is associated to a feature f_i which is defined over a subspace $[\mathcal{X}]_i \times \mathcal{Y}$ of the sample space $\mathcal{X} \times \mathcal{Y}$. Hence, $p_i(y|x)$ should be read as a shorthand of $p(y \mid [x]_i)$.
The corresponding real-valued feature is:

$$f_i(x, y) = \log p_i(y \mid x). \tag{4}$$

In this way, the ME in Eq. (3) can be rewritten as:

$$p_\lambda(y|x) = \frac{\prod_i^n p_i(y|x)^{\lambda_i}}{\sum_{y'} \prod_i p_i(y'|x)^{\lambda_i}}. \tag{5}$$

According to the formalism adopted in Eq. (4), real-valued features assume the following form:

$$f_i(c_t, c_{t-2}^{t-1}, w_{t-2}^{t+2}) = \log p_i(c_t \mid c_{t-2}^{t-1}, w_{t-2}^{t+2}). \tag{6}$$

For each so far presented type of binary feature, a corresponding real-valued type can be easily defined. The complete list is shown in Table 2. In general, the context subspace was defined on the basis of the offset parameters of each binary feature. For instance, all lexical features selecting two words at distances -1 and 0 from the current position t are modeled by the conditional distribution $p(c_t \mid w_{t-1}, w_t)$. While distributions of lexical, syntactic and transition features are conditioned on words or tags, dictionary and orthographic features are conditioned on binary variables.

An additional real-valued feature that was employed is the so called *prior feature*, i.e. the probability of a tag to occur:

$$\mathtt{Prior}(x, y) = \log p(c_t)$$

A major effect of using real-valued features is the drastic reduction of model parameters. For example, each complex lexical features discussed before introduce just one parameter. Hence, the small number of parameters eliminates the need of smoothing the ME estimates.

Real-valued features present some drawbacks. Their level of granularity, or discrimination, might result much lower than their binary variants. For many features, it might result difficult to compute reliable probability values due to data sparseness. For the last issue, smoothing techniques developed for statistical language models can be applied (Manning and Schutze, 1999).

5 Mixed Feature Models

This work, beyond investigating the use of real-valued features, addresses the behavior of models combining binary and real-valued features. The reason is twofold: on one hand, real-valued features allow to capture complex information with fewer parameters; on the other hand, binary features permit to keep a good level of granularity over salient characteristics. Hence, finding a compromise between binary and real-valued features

might help to develop ME models which better trade-off complexity vs. granularity of information.

6 Parameter Estimation

From the duality of ME and maximum likelihood (Berger et al., 1996), optimal parameters λ_* for model (3) can be found by maximizing the log-likelihood function over a training sample $\{(x_t, y_t) : t = 1, \ldots, N\}$, i.e.:

$$\lambda_* = \arg\max_{\lambda} \sum_{t=1}^{N} \log p_\lambda(y_t|x_t). \quad (7)$$

Now, whereas binary features take only two values and do not need any estimation phase, conditional probability features have to be estimated on some data sample. The question arises about how to efficiently use the available training data in order to estimate the parameters and the feature distributions of the model, by avoiding over-fitting.
Two alternative techniques, borrowed from statistical language modeling, have been considered: the *Held-out* and the *Leave-one-out* methods (Manning and Schutze, 1999).

Held-out method. The training sample $\mathcal{S}$ is split into two parts used, respectively, to estimate the feature distributions and the ME parameters.

Leave-one-out. ME parameters and feature distributions are estimated over the same sample $\mathcal{S}$. The idea is that for each addend in eq. (7), the corresponding sample point (x_t, y_t) is removed from the training data used to estimate the feature distributions of the model. In this way, it can be shown that occurrences of novel observations are simulated during the estimation of the ME parameters (Federico and Bertoldi, 2004).

In our experiments, language modeling smoothing techniques (Manning and Schutze, 1999) were applied to estimate feature distributions $p_i(y|x)$. In particular, smoothing was based on the discounting method in Ney et al. (1994) combined to interpolation with distributions using less context. Given the small number of smoothing parameters involved, leave-one-out probabilities were approximated by just modifying count statistics on the fly (Federico and Bertoldi, 2004). The rationale is that smoothing parameters do not change significantly after removing just one sample point.

For parameter estimation, the GIS algorithm by Darroch and Ratcliff (1972) was applied. It is known that the GIS algorithm requires feature functions $f_i(x, y)$ to be non-negative. Hence, features were re-scaled as follows:

$$f_i(x, y) = \log p_i(y|x) + log\frac{1+\epsilon}{\min p_i}, \quad (8)$$

where ϵ is a small positive constant and the denominator is a constant term defined by:

$$\min p_i = \min_{(x,y)\in\mathcal{S}} p_i(y|x). \quad (9)$$

The factor $(1 + \epsilon)$ was introduced to ensure that real-valued features are always positive. This condition is important to let features reflect the same behavior of the conditional distributions, which assign a positive probability to each event.

It is easy to verify that this scaling operation does not affect the original model but only impacts on the GIS calculations. Finally, a *slack* feature was introduced by the algorithm to satisfy the constraint that all features sum up to a constant value (Darroch and Ratcliff, 1972).

7 Experiments

This section presents results of ME models applied to two text-tagging tasks, Named Entity Recognition (NER) and Text Chunking (TC).

After a short introduction to the experimental framework, the detailed feature setting is presented. Then, experimental results are presented for the following contrastive conditions: binary versus real-valued features, training via held-out versus leave-one-out, atomic versus complex features.

7.1 Experimental Set-up

Named Entity Recognition English NER experiments were carried out on the CoNLL-2003 shared task[2]. This benchmark is based on texts from the Reuters Corpus which were manually annotated with parts-of-speech, chunk tags, and named entity categories. Four types of categories are defined: person, organization, location and miscellaneous, to include e.g. nations, artifacts, etc. A filler class is used for the remaining words. After including tags denoting the start of multi-word entities, a total of 9 tags results. Data are partitioned into training (200K words), development (50K words), and test (46K words) samples.

[2]Data and results in http://cnts.uia.ac.be/conll2003/ner.

Text Chunking English TC experiments were conducted on the CoNLL-2000 shared task[3]. Texts originate from the Wall Street Journal and are annotated with part-of-speech tags and chunks. The chunk set consists of 11 syntactic classes. The set of tags which also includes start-markers consists of 23 classes. Data is split into training (210K words) and test (47K words) samples.

Evaluation Tagging performance of both tasks is expressed in terms of F-score, namely the harmonic mean of precision and recall. Differences in performance have been statistically assessed with respect to precision and recall, separately, by applying a standard test on proportions, with significance levels $\alpha = 0.05$ and $\alpha = 0.1$. Henceforth, claimed differences in precision or recall will have their corresponding significance level shown in parenthesis.

7.2 Settings and Baseline Models

Feature selection and setting for ME models is an art. In these experiments we tried to use the same set of features with minor modifications across both tasks. In particular, used features and their settings are shown in Table 3.

Training of models with GIS and estimation of feature distributions used in-house developed toolkits. Performance of binary feature models was improved by smoothing features with Gaussian priors (Chen and Rosenfeld, 1999) with mean zero and standard deviation $\sigma = 4$. In general, tuning of models was carried out on a development set.

Most of the comparative experiments were performed on the NER task. Three baseline models using atomic features `Lex`, `Syn`, and `Tran` were investigated first: model BaseBin, with all binary features; model BaseReal, with all real-valued features plus the prior feature; model BaseMix, with real-valued `Lex` and binary `Tran` and `Syn`. Models BaseReal and BaseMix were trained with the held-out method. In particular, feature distributions were estimated on the training data while ME parameters on the development set.

7.3 Binary vs. Real-valued Features

The first experiment compares performance of the baseline models on the NER task. Experimental results are summarized in Table 4. Models BaseBin, BaseReal, and BaseMix achieved F-scores of 77.22, 76.84, and 77.34. Statistically meaningful differences were in terms of recall, between BaseBin and BaseReal ($\alpha = 0.1$), and between BaseMix and BaseReal ($\alpha = 0.05$).

[3]Data and results in http://cnts.uia.ac.be/conll2000/chunking.

Model ID	Num	P%	R%	F-score
BaseBin	580K	78.82	75.62	77.22
BaseReal	10	79.74	74.15	76.84
BaseMix	753	78.90	75.85	77.34

Table 4: Performance of baseline models on the NER task. Number of parameters, precision, recall, and F-score are reported for each model.

Model	Methods	P%	R%	F-score
BaseMix	Held-Out	78.90	75.85	77.34
BaseMix	L-O-O	80.64	76.40	78.46

Table 5: Performance of mixed feature models with two different training methods.

Despite models BaseMix and BaseBin perform comparably, the former has many fewer parameters, i.e. 753 against 580,000. In fact, BaseMix requires storing and estimating feature distributions, which is however performed at a marginal computational cost and off-line with respect to GIS training.

7.4 Training with Mixed Features

An experiment was conducted with the BaseMix model to compare the held-out and leave-one-out training methods. Results in terms of F-score are reported in Table 5. By applying the leave-one-out method F-score grows from 77.34 to 78.46, with a meaningful improvement in recall ($\alpha = 0.05$). With respect to models BaseBin and BaseReal, leave-one-out estimation significantly improved precision ($\alpha = 0.05$).

In terms of training time, ME models with real-valued features took significantly more GIS iterations to converge. Figures of cost per iteration and number of iterations are reported in Table 6. (Computation times are measured on a single CPU Pentium-4 2.8GHz.) Memory size of the training process is instead proportional to the number n of parameters.

7.5 Complex Features

A final set of experiments aims at comparing the baseline ME models augmented with complex features, again either binary only (model FinBin),

Feature	Index	NE Task	Chunking Task
Lex	c, w, d	$N(w) > 1, -2 \leq d \leq +2$	$-2 \leq d \leq +2$
Syn	c, T, p, d	$T \in \{\texttt{Pos}, \texttt{Chnk}\}, d = 0$	$T = \texttt{Pos}, -2 \leq d \leq +2$
Tran	c, c', d	$d = -2, -1$	$d = -2, -1$
Lex+	c, s, k, w_s^{s+k-1}	$s = -1, 0, k = 1$	$s = -1, 0\ k = 1$
Syn+	c, T, s, k, p_s^{s+k-1}	not used	$s = -1, 0\ \ k = 1$
Orth+	c, k, F, b_{-k}^{+k}	$F = \{\texttt{Cap}, \texttt{CAP}\},\ k = 2$	$F = \texttt{Cap},\ \ k = 1$
Dict+	c, k, L, b_{-k}^{+k}	$k = 3L = \{\texttt{LOC}, \texttt{PER}, \texttt{ORG}, \texttt{MISC}\}$	not used
Tran+	c, k, c_1^k	$k = 2$	$k = 2$

Table 3: Setting used for binary and real-valued features in the reported experiments.

Model	Single Iteration	Iterations	Total
BaseBin	54 sec	750	$\approx$ 11 h
BaseReal	9.6 sec	35,000	$\approx$ 93 h
BaseMix	42 sec	4,000	$\approx$ 46 h

Table 6: Computational cost of parameter estimation by different baseline models.

Model	Num	P%	R%	F-score
FinBin	673K	81.92	80.36	81.13
FinReal	19	83.58	74.03	78.07
FinMix	3K	84.34	80.38	82.31

Table 7: Results with complex features on the NER task.

Model	Num	P%	R%	F-score
FinBin	2M	91.04	91.48	91.26
FinReal	19	88.73	90.58	89.65
FinMix	6K	91.93	92.24	92.08

Table 8: Results with complex features on the TC task.

real-valued only (FinReal), or mixed (FinMix). Results are provided both for NER and TC.

This time, compared models use different feature settings. In fact, while previous experiments aimed at comparing the same features, in either real or binary form, these experiments explore alternatives to a full-fledged binary model. In particular, real-valued features are employed whose binary versions would introduce a prohibitively large number of parameters. Parameter estimation of models including real-valued features always applies the leave-one-out method.

For the NER task, model FinBin adds Orth+ and Dict+; FinReal adds Lex+, Orth+ and Dict+; and, FinMix adds real-valued Lex+ and binary-valued Orth+ and Dict+.

In the TC task, feature configurations are as follows: FinBin uses Lex, Syn, Tran, and Orth+; FinReal uses Lex, Syn, Tran, Prior, Orth+, Lex+, Syn+, Tran+; and, finally, FinMix uses binary Syn, Tran, Orth+ and real-valued Lex, Lex+, Syn+.

Performance of the models on the two tasks are reported in Table 7 and Table 8, respectively.

In the NER task, all final models outperform the baseline model. Improvements in precision and recall are all significant ($\alpha = 0.05$). Model FinMix improves precision with respect to model FinBin ($\alpha = 0.05$) and requires two order of magnitude fewer parameters.

In the TC task, the same trend is observed. Again, best performance is achieved by the model combining binary and real-valued features. In particular, all observable differences in terms of precision and recall are significant ($\alpha = 0.05$).

8 Discussion

In summary, this paper addressed improvements to ME models for text tagging applications. In particular, we showed how standard binary features from the literature can be mapped into corresponding log-probability distributions. ME training with the so-obtained real-valued features can be accomplished by combining the GIS algorithm with the leave-one-out or held-out methods.

With respect to the best performing systems at the CoNLL shared tasks, our models exploit a relatively smaller set of features and perform significantly worse. Nevertheless, performance achieved by our system are comparable with those reported by other ME-based systems taking part in the evaluations.

Extensive experiments on named-entity recog-

nition and text chunking have provided support to the following claims:

- The introduction of real-valued features drastically reduces the number of parameters of the ME model with a small loss in performance.
- The leave-one-out method is significantly more effective than the held-out method for training ME models including real-valued features.
- The combination of binary and real-valued features can lead to better ME models. In particular, state-of-the-art ME models with binary features are significantly improved by adding complex real-valued features which model long-span lexical dependencies.

Finally, the GIS training algorithm does not seem to be the optimal choice for ME models including real-valued features. Future work will investigate variants of and alternatives to the GIS algorithm. Preliminary experiments on the Base-Real model showed that training with the Simplex algorithm (Press et al., 1988) converges to similar parameter settings 50 times faster than the GIS algorithm.

9 Acknowledgments

This work was partially financed by the European Commission under the project FAME (IST-2000-29323), and by the Autonomous Province of Trento under the the FU-PAT project WebFaq.

References

O. Bender, F. J. Och, and H. Ney. 2003. Maximum entropy models for named entity recognition. In Walter Daelemans and Miles Osborne, editors, *Proceedings of CoNLL-2003*, pages 148–151. Edmonton, Canada.

A. L. Berger, S. A. Della Pietra, and V. J. Della Pietra. 1996. A Maximum Entropy Approach to Natural Language Processing. *Computational Linguistics*, 22(1):39–72.

A. Borthwick. 1999. *A Maximum Entropy approach to Named Entity Recognition*. Ph.D. thesis, Computer Science Department - New York University, New York, USA.

S. Chen and R. Rosenfeld. 1999. A Gaussian prior for smoothing maximum entropy models. Technical Report CMUCS-99-108, Carnegie Mellon University.

J.N. Darroch and D. Ratcliff. 1972. Generalized Iterative Scaling for Log-Liner models. *Annals of Mathematical Statistics*, 43:1470–1480.

M. Federico and N. Bertoldi. 2004. Broadcast news lm adaptation over time. *Computer Speech and Language*, 18(4):417–435, October.

D. Klakow. 1998. Log-linear interpolation of language models. In *Proceedings of the International Conference of Spoken Language P rocessing (ICSLP)*, Sidney, Australia.

R. Koeling. 2000. Chunking with maximum entropy models. In *Proceedings of CoNLL-2000*, pages 139–141, Lisbon, Portugal.

C. D. Manning and H. Schutze. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press.

A. Mikheev. 1998. Feature lattices for maximum entropy modelling. In *COLING-ACL*, pages 848–854.

H. Ney, U. Essen, and R. Kneser. 1994. On structuring probabilistic dependences in stochastic language modeling. *Computer Speech and Language*, 8(1):1–38.

F.J. Och and H. Ney. 2002. Discriminative training and maximum entropy models for statistical machin e translation. In *ACL02: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 295–302, PA, Philadelphia.

W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. 1988. *Numerical Recipes in C*. Cambridge University Press, New York, NY.

Constraint Satisfaction Inference: Non-probabilistic Global Inference for Sequence Labelling

Sander Canisius and **Antal van den Bosch**
ILK / Language and Information Science
Tilburg University
Tilburg, The Netherlands
{S.V.M.Canisius,Antal.vdnBosch@uvt.nl}@uvt.nl

Walter Daelemans
CNTS, Department of Linguistics
University of Antwerp
Antwerp, Belgium
Walter.Daelemans@ua.ac.be

Abstract

We present a new method for performing sequence labelling based on the idea of using a machine-learning classifier to generate several possible output sequences, and then applying an inference procedure to select the best sequence among those. Most sequence labelling methods following a similar approach require the base classifier to make probabilistic predictions. In contrast, our method can be used with virtually any type of classifier. This is illustrated by implementing a sequence classifier on top of a (non-probabilistic) memory-based learner. In a series of experiments, this method is shown to outperform two other methods; one naive baseline approach, and another more sophisticated method.

1 Introduction

In machine learning for natural language processing, many diverse tasks somehow involve processing of sequentially-structured data. For example, syntactic chunking, grapheme-to-phoneme conversion, and named-entity recognition are all usually reformulated as sequence labelling tasks: a task-specific global unit, such as a sentence or a word, is divided into atomic sub-parts, e.g. word or letters, each of which is separately assigned a label. The concatenation of those labels forms the eventual output for the global unit.

More formally, we can define a sequence labelling task as a tuple $(\mathbf{x}, \mathbf{y}, \ell)$. The goal is to map an input vector $\mathbf{x} = \langle x_1, x_2, \ldots, x_n \rangle$ of tokens to an output sequence $\mathbf{y} = \langle y_1, y_2, \ldots, y_n \rangle$ of labels. The possible labels for each token are specified by a finite set ℓ, that is, $y_i \in \ell, \forall i$.

In most real-world sequence labelling tasks, the values of the output labels are sequentially correlated. For machine learning approaches to sequence labelling this implies that classifying each token separately without considering the labels assigned to other tokens in the sequence may lead to sub-optimal performance. Ideally, the complex mapping of the entire input sequence to its corresponding output sequence is considered one classification case; the classifier then has access to all information stored in the sequence. In practise, however, both input and output sequences are far too sparse for such classifications to be performed reliably.

A popular approach to circumvent the issues raised above is what we will refer to as the classification and inference approach, covering techniques such as hidden markov models and conditional random fields (Lafferty et al., 2001). Rather than having a token-level classifier make local decisions independently of the rest of the sequence, the approach introduces an inference procedure, operating on the level of the sequence, using class likelihoods estimated by the classifier to optimise the likelihood of the entire output sequence.

A crucial property of most of the classification and inference techniques in use today is that the classifier used at the token level must be able to estimate the likelihood for each potential class label. This is in contrast with the more common view of a classifier having to predict just one class label for an instance which is deemed most optimal. Maximum-entropy models, which are used in many classification and inference techniques, have this property; they model the conditional class distribution. In general, this is the case for all probabilistic classification methods. However, many general-purpose machine learning techniques are

not probabilistic. In order to design inference procedures for those techniques, other principles than probabilistic ones have to be used.

In this paper, we propose a non-probabilistic inference procedure that improves performance of a memory-based learner on a wide range of natural-language sequence processing tasks. We start from a technique introduced recently by Van den Bosch and Daelemans (2005), and reinterpret it as an instance of the classification and inference approach. Moreover, the token-level inference procedure proposed in the original work is replaced by a new procedure based on principles of constraint satisfaction that does take into account the entire sequential context.

The remainder of this paper is structured as follows. Section 2 introduces the theoretical background and starting point of the work presented in this paper: the trigram method, and memory-based learning. Next, the new constraint-satisfaction-based inference procedure for class trigrams is presented in Section 3. Experimental comparisons of a non-sequence-aware baseline classifier, the original trigram method, and the new classification and inference approach on a number of sequence labelling tasks are presented in Section 4 and discussed in Section 5. Finally, our work is compared and contrasted with some related approaches in Section 6, and conclusions are drawn in Section 7.

2 Theoretical background

2.1 Class Trigrams

A central weakness of approaches considering each token of a sequence as a separate classification case is their inability to coordinate labels assigned to neighbouring tokens. Due to this, invalid label sequences, or ones that are highly unlikely may result. Van den Bosch and Daelemans (2005) propose to resolve parts of this issue by predicting trigrams of labels as a single atomic class label, thereby labelling three tokens at once, rather than classifying each token separately. Predicting sequences of three labels at once makes sure that at least these short subsequences are known to be syntactically valid sequences according to the training data.

Applying this general idea, Van den Bosch and Daelemans (2005) label each token with a complex class label composed of the labels for the preceding token, the token itself, and the one following it in the sequence. If such class trigrams are assigned to all tokens in a sequence, the actual label for each of those is effectively predicted three times, since every token but the first and last is covered by three class trigrams. Exploiting this redundancy, a token's possibly conflicting predictions are resolved by voting over them. If two out of three trigrams suggest the same label, this label is selected; in case of three different candidate labels, a classifier-specific confidence metric is used to break the tie.

Voting over class trigrams is but one possible approach to taking advantage of the redundancy obtained with predicting overlapping trigrams. A disadvantage of voting is that it discards one of the main benefits of the class trigram method: predicted class trigrams are guaranteed to be syntactically correct according to the training data. The voting technique splits up the predicted trigrams, and only refers to their unigram components when deciding on the output label for a token; no attempt is made to keep the trigram sequence intact in the final output sequence. The alternative to voting presented later in this paper does try to retain predicted trigrams as part of the output sequence.

2.2 Memory-based learning

The name memory-based learning refers to a class of methods based on the k-nearest neighbour rule. At training time, all example instances are stored in memory without attempting to induce an abstract representation of the concept to be learned. Generalisation is postponed until a test instance is classified. For a given test instance, the class predicted is the one observed most frequently among a number of most-similar instances in the instance base. By only generalising when confronted with the instance to be classified, a memory-based learner behaves as a local model, specifically suited for that part of the instance space that the test instance belongs to. In contrast, learners that abstract at training time can only generalise globally. This distinguishing property makes memory-based learners especially suited for tasks where different parts of the instance space are structured according to different rules, as is often the case in natural-language processing.

For the experiments performed in this study we used the memory-based classifier as implemented by TiMBL (Daelemans et al., 2004). In TiMBL, similarity is defined by two parameters: a feature-level similarity metric, which assigns a real-valued

score to pairs of values for a given feature, and a set of feature weights, that express the importance of the various features for determining the similarity of two instances. Further details on both of these parameters can be found in the TiMBL manual. To facilitate the explanation of our inference procedure in Section 3, we will formally define some notions related to memory-based classification.

The function $N_{s,w,k}(x)$ maps a given instance x to the set of its nearest neighbours; here, the parameters s, w, and k are the similarity metric, the feature weights, and the number k of nearest neighbours, respectively. They will be considered given in the following, so we will refer to this specific instantiation simply as $N(x)$. The function $w_d(c, N(x))$ returns the weight assigned to class c in the given neighbourhood according to the distance metric d; again we will use the notation $w(c, N(x))$ to refer to a specific instantiation of this function. Using these two functions, we can formulate the nearest neighbour rule as follows.

$$\arg\max_c w(c, N(x))$$

The class c maximising the above expression is returned as the predicted class for the instance x.

3 Constraint Satisfaction Inference

A strength of the class trigram method is the guarantee that any trigram that is predicted by the base classifier represents a syntactically valid subsequence of length three. This does not necessarily mean the trigram is a correct label assignment within the context of the current classification, but it does reflect the fact that the trigram has been observed in the training data, and, moreover, is deemed most likely according to the base classifier's model. For this reason, it makes sense to try to retain predicted trigrams in the output label sequence as much as possible.

The inference method proposed in this section seeks to attain this goal by formulating the class trigram disambiguation task as a weighted constraint satisfaction problem (W-CSP). Constraint satisfaction is a well-studied research area with applications in numerous fields both inside and outside of computer science. Weighted constraint satisfaction extends the traditional constraint satisfaction framework with soft constraints; such constraints are not required to be satisfied for a solution to be valid, but constraints a given solution does satisfy, are rewarded according to weights assigned to them.

Formally, a W-CSP is a tuple (X, D, C, W). Here, $X = \{x_1, x_2, \ldots, x_n\}$ is a finite set of variables. $D(x)$ is a function that maps each variable to its domain, that is, the set of values that variable can take on. C is the set of constraints. While a variable's domain dictates the values a single variable is allowed to take on, a constraint specifies which simultaneous value *combinations* over a number of variables are allowed. For a traditional (non-weighted) constraint satisfaction problem, a valid solution would be an assignment of values to the variables that (1) are a member of the corresponding variable's domain, and (2) satisfy *all* constraints in the set C. Weighted constraint satisfaction, however, relaxes this requirement to satisfy all constraints. Instead, constraints are assigned weights that may be interpreted as reflecting the importance of satisfying that constraint.

Let a constraint $c \in C$ be defined as a function that maps each variable assignment to 1 if the constraint is satisfied, or to 0 if it is not. In addition, let W: $C \rightarrow \mathbb{R}^+$ denote a function that maps each constraint to a positive real value, reflecting the weight of that constraint. Then, the optimal solution to a W-CSP is given by the following equation.

$$\mathbf{x}^* = \arg\max_{\mathbf{x}} \sum_c W(c)c(\mathbf{x})$$

That is, the assignment of values to its variables that maximises the sum of weights of the constraints that have been satisfied.

Translating the terminology introduced earlier in this paper to the constraint satisfaction domain, each token of a sequence maps to a variable, the domain of which corresponds to the three candidate labels for this token suggested by the trigrams covering the token. This provides us with a definition of the function D, mapping variables to their domain. In the following, $y_{i,j}$ denotes the candidate label for token x_j predicted by the trigram assigned to token x_i.

$$D(x_i) = \{y_{i-1,i}, y_{i,i}, y_{i+1,i}\}$$

Constraints are extracted from the predicted trigrams. Given the goal of retaining predicted trigrams in the output label sequence as much as possible, the most important constraints are simply

the trigrams themselves. A predicted trigram describes a subsequence of length three of the entire output sequence; by turning such a trigram into a constraint, we express the wish to have this trigram end up in the final output sequence.

$$(x_{i-1}, x_i, x_{i+1}) = (y_{i,i-1}, y_{i,i}, y_{i,i+1}), \forall i$$

No base classifier is flawless though, and therefore not all predicted trigrams can be expected to be correct. Nevertheless, even an incorrect trigram may carry some useful information regarding the output sequence: one trigram also covers two bigrams, and three unigrams. An incorrect trigram may still contain smaller subsequences, of length one or two, that are correct. Therefore, all of these are also mapped to constraints.

$$\begin{aligned}(x_{i-1}, x_i) &= (y_{i,i-1}, y_{i,i}), && \forall i \\ (x_i, x_{i+1}) &= (y_{i,i}, y_{i,i+1}), && \forall i\end{aligned}$$

$$\begin{aligned}x_{i-1} &= y_{i,i-1}, && \forall i \\ x_i &= y_{i,i}, && \forall i \\ x_{i+1} &= y_{i,i+1}, && \forall i\end{aligned}$$

With such an amount of overlapping constraints, the satisfaction problem obtained easily becomes over-constrained, that is, no variable assignment exists that can satisfy all constraints without breaking another. Only one incorrectly predicted class trigram already leads to two conflicting candidate labels for one of the tokens at least. Yet, without conflicting candidate labels no inference would be needed to start with. The choice for the weighted constraint satisfaction method always allows a solution to be found, even in the presence of conflicting constraints. Rather than requiring all constraints to be satisfied, each constraint is assigned a certain weight; the optimal solution to the problem is an assignment of values to the variables that optimises the sum of weights of the constraints that are satisfied.

Constraints can directly be traced back to a prediction made by the base classifier. If two constraints are in conflict, the one which the classifier was most certain of should preferably be satisfied. In the W-CSP framework, this preference can be expressed by weighting constraints according to the classifier confidence for the originating trigram. For the memory-based learner, we define the classifier confidence for a predicted class c_i as the weight assigned to that class in the neighbourhood of the test instance, divided by the total weight of all classes.

$$\frac{w(c_i, N(x))}{\sum_c w(c, N(x))}$$

Let x denote a test instance, and c^* its predicted class. Constraints derived from this class are weighted according to the following rules.

- for a trigram constraint, the weight is simply the base classifier's confidence value for the class c^*
- for a bigram constraint, the weight is the sum of the confidences for all trigram classes in the nearest-neighbour set of x that assign the same label bigram to the tokens spanned by the constraint
- for a unigram constraint, the weight is the sum of the confidences for all trigram classes in the nearest-neighbour set of x that assign the same label to the token spanned by the constraint

4 Experiments

To thoroughly evaluate our new inference procedure, and to show that it performs well over a wide range of natural-language sequence labelling tasks, we composed a benchmark set consisting of six different tasks, covering four areas in natural language processing: syntax (syntactic chunking), morphology (morphological analysis), phonology (grapheme-to-phoneme conversion), and information extraction (general, medical, and biomedical named-entity recognition). Below, the six data sets used for these tasks are introduced briefly.

CHUNK is the task of splitting sentences into non-overlapping syntactic phrases or constituents. The data set, extracted from the WSJ Penn Treebank, and first used in the CoNLL-2000 shared task (Tjong Kim Sang and Buchholz, 2000), contains 211,727 training examples and 47,377 test instances.

NER, named-entity recognition, involves identifying and labelling named entities in text. We employ the English NER shared task data set used in the CoNLL-2003 conference (Tjong Kim Sang and De Meulder, 2003). This data set discriminates four name types: persons, organisations, locations, and a rest category of "miscellany names". The data set is a collection of newswire

articles from the Reuters Corpus, RCV1[1]. The given training set contains 203,621 examples; as test set we use the "testb" evaluation set which contains 46,435 examples.

MED is a data set extracted from a semantic annotation of (parts of) two Dutch-language medical encyclopedias. On the chunk-level of this annotation, there are labels for various medical concepts, such as disease names, body parts, and treatments, forming a set of twelve concept types in total. Chunk sizes range from one to a few tokens. The data have been split into training and test sets, resulting in 428,502 training examples and 47,430 test examples.

The GENIA corpus (Tateisi et al., 2002) is a collection of annotated abstracts taken from the National Library of Medicine's MEDLINE database. Apart from part-of-speech tagging information, the corpus annotates a subset of the substances and the biological locations involved in reactions of proteins. Using a 90%–10% split for producing training and test sets, there are 458,593 training examples and 50,916 test examples.

PHONEME refers to grapheme-to-phoneme conversion for English. The sequences to be labelled are words composed of letters (rather than sentences composed of words). We based ourselves on the English part of the CELEX-2 lexical data base (Baayen et al., 1993), from which we extracted 65,467 word-pronunciation pairs. This pair list has been aligned using expectation-maximisation to obtain sensible one-to-one mappings between letters and phonemes (Daelemans and Van den Bosch, 1996). The classes to predict are 58 different phonemes, including some diphones such as [ks] needed to keep the letter-phoneme alignment one-to-one. The resulting data set has been split into a training set of 515,891 examples, and a test set of 57,279 examples.

MORPHO refers to morphological analysis of Dutch words. We collected the morphological analysis of 336,698 Dutch words from the CELEX-2 lexical data base (Baayen et al., 1993), and represented the task such that it captures the three most relevant elements of a morphological analysis: (1) the segmentation of the word into morphemes (stems, derivational morphemes, and inflections), (2) the part-of-speech tagging information contained by each morpheme; and (3) all spelling changes due to compounding, derivation, or inflection that would enable the reconstruction of the appropriate root forms of the involved morphemes.

[1]Reuters Corpus, Volume 1, English language, 1996-08-20 to 1997-08-19.

Task	Baseline	Voting	CSInf	Oracle
CHUNK	91.9	92.7	93.1	95.8
NER	77.2	80.2	81.8	86.5
MED	64.7	67.5	68.9	74.9
GENIA	55.8	60.1	61.8	70.6
PHONEME	79.0	83.4	84.5	98.8
MORPHO	41.3	46.1	51.9	62.2

Table 1: Performances of the baseline method, and the trigram method combined both with majority voting, and with constraint satisfaction inference. The last column shows the performance of the (hypothetical) oracle inference procedure.

For CHUNK, and the three information extraction tasks, instances represent a seven-token window of words and their (predicted) part-of-speech tags. Each token is labelled with a class using the IOB type of segmentation coding as introduced by Ramshaw and Marcus (1995), marking whether the middle word is inside (I), outside (O), or at the beginning (B) of a chunk, or named entity. Performance is measured by the F-score on correctly identified and labelled chunks, or named entities.

Instances for PHONEME, and MORPHO consist of a seven-letter window of letters only. The labels assigned to an instance are task-specific and have been introduced above, together with the tasks themselves. Generalisation performance is measured on the word accuracy level: if the entire phonological transcription of the word is predicted correctly, or if all three aspects of the morphological analysis are predicted correctly, the word is counted correct.

4.1 Results

For the experiments, memory-based learners were trained and automatically optimised with wrapped progressive sampling (Van den Bosch, 2004) to predict class trigrams for each of the six tasks introduced above. Table 1 lists the performances of constraint satisfaction inference, and majority voting applied to the output of the base classifiers, and compares them with the performance of a naive baseline method that treats each token as a separate classification case without coordinating decisions over multiple tokens.

Without exception, constraint satisfaction infer-

ence outperforms majority voting by a considerable margin. This shows that, given the same sequence of predicted trigrams, the global constraint satisfaction inference manages better to recover sequential correlation, than majority voting. On the other hand, the error reduction attained by majority voting with respect to the baseline is in all cases more impressive than the one obtained by constraint satisfaction inference with respect to majority voting. However, it should be emphasised that, while both methods trace back their origins to the work of Van den Bosch and Daelemans (2005), constraint satisfaction inference is not applied after, but instead of majority voting. This means, that the error reduction attained by majority voting is also attained, independently by constraint satisfaction inference, but in addition constraint satisfaction inference manages to improve performance on top of that.

5 Discussion

The experiments reported upon in the previous section showed that by globally evaluating the quality of possible output sequences, the constraint satisfaction inference procedure manages to attain better results than the original majority voting approach. In this section, we attempt to further analyse the behaviour of the inference procedure. First, we will discuss the effect that the performance of the trigram-predicting base classifier has on the maximum performance attainable by any inference procedure. Next, we will consider specifically the effect of base classifier accuracy on the performance of constraint satisfaction inference.

5.1 Base classifier accuracy and inference procedure upper-bounds

After trigrams have been predicted, for each token, at most three different candidate labels remain. As a result, if the correct label is not among them, the best inference procedure cannot correct that. This suggests that there is an upper-bound on the performance attainable by inference procedures operating on less than perfect class trigram predictions. To illustrate what performance is still possible after a base classifier has predicted the trigrams for a sequence, we devised an oracle inference procedure.

An oracle has perfect knowledge about the true label of a token; therefore it is able to select this label if it is among the three candidate labels. If the correct label is absent among the candidate labels, no inference procedure can possibly predict the correct label for the corresponding token, so the oracle procedure just selects randomly among the candidate labels, which will be incorrect anyway. Table 1 compares the performance of majority voting, constraint satisfaction inference, and the oracle after an optimised base classifier has predicted class trigrams.

5.2 Base classifier accuracy and constraint satisfaction inference performance

There is a subtle balance between the quality of the trigram-predicting base classifier, and the gain that any inference procedure for trigram classes can reach. If the base classifier's predictions are perfect, all three candidate labels will agree for all tokens in the sequence; consequently the inference procedure can only choose from one potential output sequence. On the other extreme, if all three candidate labels disagree for all tokens in the sequence, the inference procedure's task is to select the best sequence among 3^n possible sequences, where n denotes the length of the sequence; it is likely that such a huge amount of candidate label sequences cannot be dealt with appropriately.

Table 2 collects the base classifier accuracies, and the average number of potential output sequences per sentence resulting from its predictions. For all tasks, the number of potential sequences is manageable; far from the theoretical maximum 3^n, even for GENIA, that, compared with the other tasks, has a relatively large number of potential output sequences. The factors that have an effect on the number of sequences are rather complex. One important factor is the accuracy of the trigram predictions made by the base classifier. To illustrate this, Figure 1 shows the number of potential output sequences as a function of the base classifier accuracy for the PHONEME task. There is an almost linear decrease of the number of possible sequences as the classifier accuracy improves. This shows that it is important to optimise the performance of the base classifier, since it decreases the number of potential output sequences to consider for the inference procedure. Other factors affecting the number of potential output sequences are the length of the sequence, and the number of labels defined for the task. Unlike classifier accuracy, however, these two factors

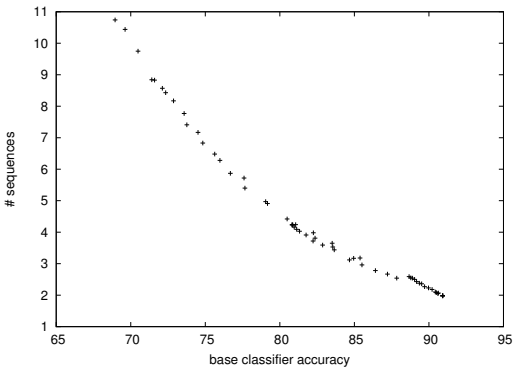


Figure 1: Average number of potential output sequences as a function of base classifier accuracy on the PHONEME task.

Task	Base acc.	Avg. # seq.
CHUNK	88.8	38.4
NER	91.6	9.0
MED	77.1	9.3
GENIA	71.8	1719.3
PHONEME	91.7	1.8
MORPHO	80.9	2.8

Table 2: The average number of potential output sequences that result from class trigram predictions made by a memory-based base classifier.

are inherent properties of the task, and cannot be optimised.

While we have shown that improved base classifier accuracy has a positive effect on the number of possible output sequences; we have not yet established a positive relation between the number of possible output sequences and the performance of constraint satisfaction inference. Figure 2 illustrates, again for the PHONEME task, that there is indeed a positive, even linear relation between the accuracy of the base classifier, and the performance attained by inference. This relation exists for all inference procedures: majority voting, as well as constraint satisfaction inference, and the oracle procedure. It is interesting to see how the curves for those three procedure compare with each other.

The oracle always outperforms the other two procedures by a wide margin. However, its increase is less steep. Constraint satisfaction inference consistently outperforms majority voting, though the difference between the two decreases as the base classifier's predictions improve. This is to be expected, since more accurate predictions means more majorities will appear among candi-

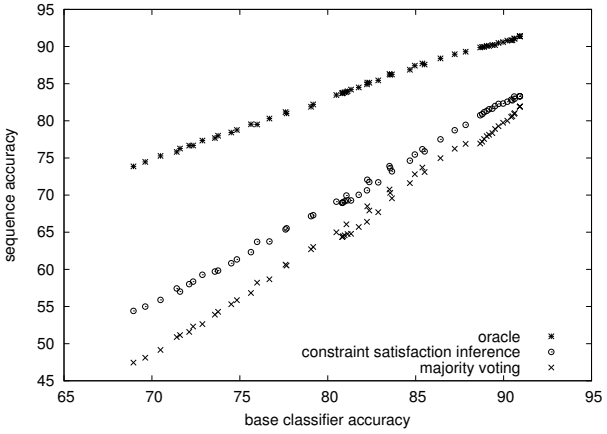


Figure 2: Performance of majority voting, constraint satisfaction inference, and the oracle inference procedure as a function of base classifier accuracy on the PHONEME task.

date labels, and the predictive quality of such majorities improves as well. In the limit –with a perfect base classifier– all three curves will meet.

6 Related work

Many learning techniques specifically designed for sequentially-structured data exist. Given our goal of developing a method usable with non-probabilistic classifiers, we will not discuss the obvious differences with the many probabilistic methods. In this section, we will contrast our work with two other approaches that also apply principles of constraint satisfaction to sequentially-structured data.

Constraint Satisfaction with Classifiers (Punyakanok and Roth, 2001) performs the somewhat more specific task of identifying phrases in a sequence. Like our method, the task of coordinating local classifier decisions is formulated as a constraint satisfaction problem. The variables encode whether or not a certain contiguous span of tokens forms a phrase. Hard constraints enforce that no two phrases in a solution overlap.

Similarly to our method, classifier confidence estimates are used to rank solutions in order of preference. Unlike in our method, however, both the domains of the variables and the constraints are prespecified; the classifier is used only to estimate the cost of potential variable assignments. In our approach, the classifier predicts the domains of the variables, the constraints, and the weights of those.

Roth and Yih (2005) replace the Viterbi algo-

rithm for inference in conditional random fields with an integer linear programming formulation. This allows arbitrary global constraints to be incorporated in the inference procedure. Essentially, the method adds constraint satisfaction functionality on top of the inference procedure. In our method, constraint satisfaction *is* the inference procedure. Nevertheless, arbitrary global constraints (both hard and soft) can easily be incorporated in our framework as well.

7 Conclusion

The classification and inference approach is a popular and effective framework for performing sequence labelling in tasks where there is strong interaction between output labels. Most existing inference procedures expect a base classifier that makes probabilistic predictions, that is, rather than predicting a single class label, a conditional probability distribution over the possible classes is computed. The inference procedure presented in this paper is different in the sense that it can be used with any classifier that is able to estimate a confidence score for its (non-probabilistic) predictions.

Constraint satisfaction inference builds upon the class trigram method introduced by Van den Bosch and Daelemans (2005), but reinterprets it as a strategy for generating multiple potential output sequences, from which it selects the sequence that has been found to be most optimal according to a weighted constraint satisfaction formulation of the inference process. In a series of experiments involving six sequence labelling task covering several different areas in natural language processing, constraint satisfaction inference has been shown to improve substantially upon the performance achieved by a simpler inference procedure based on majority voting, proposed in the original work on the class trigram method.

The work presented in this paper shows there is potential for alternative interpretations of the classification and inference framework that do not rely on probabilistic base classifiers. Future work may well be able to further improve the performance of constraint satisfaction inference, for example, by using more optimised constraint weighting schemes. In addition, alternative ways of formulating constraint satisfaction problems from classifier predictions may be explored; not only for sequence labelling, but also for other domains that could benefit from global inference.

References

R. H. Baayen, R. Piepenbrock, and H. van Rijn. 1993. *The CELEX lexical data base on CD-ROM*. Linguistic Data Consortium, Philadelphia, PA.

W. Daelemans and A. Van den Bosch. 1996. Language-independent data-oriented grapheme-to-phoneme conversion. In J. P. H. Van Santen, R. W. Sproat, J. P. Olive, and J. Hirschberg, editors, *Progress in Speech Processing*, pages 77–89. Springer-Verlag, Berlin.

W. Daelemans, J. Zavrel, K. Van der Sloot, and A. Van den Bosch. 2004. TiMBL: Tilburg memory based learner, version 5.1.0, reference guide. Technical Report ILK 04-02, ILK Research Group, Tilburg University.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fi elds: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*, Williamstown, MA.

V. Punyakanok and D. Roth. 2001. The use of classifi ers in sequential inference. In *NIPS-13; The 2000 Conference on Advances in Neural Information Processing Systems*, pages 995–1001. The MIT Press.

L.A. Ramshaw and M.P. Marcus. 1995. Text chunking using transformation-based learning. In *Proceedings of the 3rd ACL/SIGDAT Workshop on Very Large Corpora, Cambridge, Massachusetts, USA*, pages 82–94.

D. Roth and W. Yih. 2005. Integer linear programming inference for conditional random fi elds. In *Proc. of the International Conference on Machine Learning (ICML)*, pages 737–744.

Yuka Tateisi, Hideki Mima, Ohta Tomoko, and Junichi Tsujii. 2002. Genia corpus: an annotated research abstract corpus in molecular biology domain. In *Human Language Technology Conference (HLT 2002)*, pages 73–77.

E. Tjong Kim Sang and S. Buchholz. 2000. Introduction to the CoNLL-2000 shared task: Chunking. In *Proceedings of CoNLL-2000 and LLL-2000*, pages 127–132.

E. Tjong Kim Sang and F. De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In W. Daelemans and M. Osborne, editors, *Proceedings of CoNLL-2003*, pages 142–147. Edmonton, Canada.

A. Van den Bosch and W. Daelemans. 2005. Improving sequence segmentation learning by predicting trigrams. In I. Dagan and D. Gildea, editors, *Proceedings of the Ninth Conference on Computational Natural Language Learning*.

A. Van den Bosch. 2004. Wrapped progressive sampling search for optimizing learning algorithm parameters. In R. Verbrugge, N. Taatgen, and L. Schomaker, editors, *Proceedings of the 16th Belgian-Dutch Conference on Artificial Intelligence*, pages 219–226, Groningen, The Netherlands.

Decomposition Kernels for Natural Language Processing

Fabrizio Costa **Sauro Menchetti** **Alessio Ceroni** **Andrea Passerini** **Paolo Frasconi**
Dipartimento di Sistemi e Informatica,
Università degli Studi di Firenze,
via di S. Marta 3, 50139 Firenze, Italy
{costa,menchett,passerini,aceroni,p-f} AT dsi.unifi.it

Abstract

We propose a simple solution to the sequence labeling problem based on an extension of weighted decomposition kernels. We additionally introduce a multi-instance kernel approach for representing lexical word sense information. These new ideas have been preliminarily tested on named entity recognition and PP attachment disambiguation. We finally suggest how these techniques could be potentially merged using a declarative formalism that may provide a basis for the integration of multiple sources of information when using kernel-based learning in NLP.

1 Introduction

Many tasks related to the analysis of natural language are best solved today by machine learning and other data driven approaches. In particular, several subproblems related to information extraction can be formulated in the supervised learning framework, where statistical learning has rapidly become one of the preferred methods of choice. A common characteristic of many NLP problems is the relational and structured nature of the representations that describe data and that are internally used by various algorithms. Hence, in order to develop effective learning algorithms, it is necessary to cope with the inherent structure that characterize linguistic entities. Kernel methods (see e.g. Shawe-Taylor and Cristianini, 2004) are well suited to handle learning tasks in structured domains as the statistical side of a learning algorithm can be naturally decoupled from any representational details that are handled by the kernel function. As a matter of facts, kernel-based statistical learning has gained substantial importance in the NLP field. Applications are numerous and diverse and include for example refinement of statistical parsers (Collins and Duffy, 2002), tagging named entities (Cumby and Roth, 2003; Tsochantaridis et al., 2004), syntactic chunking (Daumé III and Marcu, 2005), extraction of relations between entities (Zelenko et al., 2003; Culotta and Sorensen, 2004), semantic role labeling (Moschitti, 2004). The literature is rich with examples of kernels on discrete data structures such as sequences (Lodhi et al., 2002; Leslie et al., 2002; Cortes et al., 2004), trees (Collins and Duffy, 2002; Kashima and Koyanagi, 2002), and annotated graphs (Gärtner, 2003; Smola and Kondor, 2003; Kashima et al., 2003; Horváth et al., 2004). Kernels of this kind can be almost invariably described as special cases of convolution and other decomposition kernels (Haussler, 1999). Thanks to its generality, decomposition is an attractive and flexible approach for defining the similarity between structured objects starting from the similarity between smaller parts. However, excessively large feature spaces may result from the combinatorial growth of the number of distinct subparts with their size. When too many dimensions in the feature space are irrelevant, the Gram matrix will be nearly diagonal (Schölkopf et al., 2002), adversely affecting generalization in spite of using large margin classifiers (Ben-David et al., 2002). Possible cures include extensive use of prior knowledge to guide the choice of relevant parts (Cumby and Roth, 2003; Frasconi et al., 2004), the use of feature selection (Suzuki et al., 2004), and soft matches (Saunders et al., 2002). In (Menchetti et al., 2005) we have shown that better generalization can indeed be achieved by avoiding hard comparisons between large parts. In a

weighted decomposition kernel (WDK) only small parts are matched, whereas the importance of the match is determined by comparing the sufficient statistics of elementary probabilistic models fitted on larger contextual substructures. Here we introduce a position-dependent version of WDK that can solve sequence labeling problems without searching the output space, as required by other recently proposed kernel-based solutions (Tsochantaridis et al., 2004; Daumé III and Marcu, 2005).

The paper is organized as follows. In the next two sections we briefly review decomposition kernels and its weighted variant. In Section 4 we introduce a version of WDK for solving supervised sequence labeling tasks and report a preliminary evaluation on a named entity recognition problem. In Section 5 we suggest a novel multi-instance approach for representing WordNet information and present an application to the PP attachment ambiguity resolution problem. In Section 6 we discuss how these ideas could be merged using a declarative formalism in order to integrate multiple sources of information when using kernel-based learning in NLP.

2 Decomposition Kernels

An R-decomposition structure (Haussler, 1999; Shawe-Taylor and Cristianini, 2004) on a set $\mathcal{X}$ is a triple $\mathcal{R} = \langle \vec{\mathcal{X}}, R, \vec{k} \rangle$ where $\vec{\mathcal{X}} = (\mathcal{X}_1, \ldots, \mathcal{X}_D)$ is a D–tuple of non–empty subsets of $\mathcal{X}$, R is a finite relation on $\mathcal{X}_1 \times \cdots \times \mathcal{X}_D \times \mathcal{X}$, and $\vec{k} = (k_1, \ldots, k_D)$ is a D–tuple of positive definite kernel functions $k_d : \mathcal{X}_d \times \mathcal{X}_d \mapsto \mathbb{R}$. $R(\vec{x}, x)$ is true iff $\vec{x}$ is a tuple of "parts" for x — i.e. $\vec{x}$ is a decomposition of x. Note that this definition of "parts" is very general and does not require the parthood relation to obey any specific mereological axioms, such as those that will be introduced in Section 6. For any $x \in \mathcal{X}$, let $R^{-1}(x) = \{(x_1, \ldots, x_D) \in \vec{\mathcal{X}} : R(\vec{x}, x)\}$ denote the multiset of all possible decompositions[1] of x. A decomposition kernel is then defined as the *multiset kernel* between the decompositions:

$$K_{\mathcal{R}}(x, x') = \sum_{\substack{\vec{x} \in R^{-1}(x) \\ \vec{x}' \in R^{-1}(x')}} \prod_{d=1}^{D} \kappa_d(x_d, x'_d) \qquad (1)$$

[1]Decomposition examples in the string domain include taking all the contiguous fixed-length substrings or all the possible ways of dividing a string into two contiguous substrings.

where, as an alternative way of combining the kernels, we can use the product instead of a summation: intuitively this increases the feature space dimension and makes the similarity measure more selective. Since decomposition kernels form a rather vast class, the relation R needs to be carefully tuned to different applications in order to characterize a suitable kernel. As discussed in the Introduction, however, taking *all* possible subparts into account may lead to poor predictivity because of the combinatorial explosion of the feature space.

3 Weighted Decomposition Kernels

A weighted decomposition kernel (WDK) is characterized by the following decomposition structure:

$$\mathcal{R} = \langle \vec{\mathcal{X}}, R, (\delta, \kappa_1, \ldots, \kappa_D) \rangle$$

where $\vec{\mathcal{X}} = (S, Z_1, \ldots, Z_D)$, $R(s, z_1, \ldots, z_D, x)$ is true iff $s \in S$ is a subpart of x called the *selector* and $\vec{z} = (z_1, \ldots, z_D) \in Z_1 \times \cdots \times Z_D$ is a tuple of subparts of x called the *contexts* of s in x. Precise definitions of s and $\vec{z}$ are domain-dependent. For example in (Menchetti et al., 2005) we present two formulations, one for comparing whole sequences (where both the selector and the context are subsequences), and one for comparing attributed graphs (where the selector is a single vertex and the context is the subgraph reachable from the selector within a short path). The definition is completed by introducing a kernel on selectors and a kernel on contexts. The former can be chosen to be the exact matching kernel, δ, on $S \times S$, defined as $\delta(s, s') = 1$ if $s = s'$ and $\delta(s, s') = 0$ otherwise. The latter, κ_d, is a kernel on $Z_d \times Z_d$ and provides a soft similarity measure based on attribute frequencies. Several options are available for context kernels, including the discrete version of probability product kernels (PPK) (Jebara et al., 2004) and histogram intersection kernels (HIK) (Odone et al., 2005). Assuming there are n categorical attributes, each taking on m_i distinct values, the context kernel can be defined as:

$$\kappa_d(z, z') \;=\; \sum_{i=1}^{n} k_i(z, z') \qquad (2)$$

where k_i is a kernel on the i-th attribute. Denote by $p_i(j)$ the observed frequency of value j in z. Then

k_i can be defined as a HIK or a PPK respectively:

$$k_i(z, z') = \sum_{j=1}^{m_i} \min\{p_i(j), p'_i(j)\} \quad (3)$$

$$k_i(z, z') = \sum_{j=1}^{m_i} \sqrt{p_i(j) \cdot p'_i(j)} \quad (4)$$

This setting results in the following general form of the kernel:

$$K(x, x') = \sum_{\substack{(s,\vec{z}) \in R^{-1}(x) \\ (s',\vec{z}') \in R^{-1}(x')}} \delta(s, s') \sum_{d=1}^{D} \kappa_d(z_d, z'_d) \quad (5)$$

where we can replace the summation of kernels with $\prod_{d=1}^{D} 1 + \kappa_d(z_d, z'_d)$.

Compared to kernels that simply count the number of substructures, the above function weights different matches between selectors according to contextual information. The kernel can be afterwards normalized in $[-1, 1]$ to prevent similarity to be boosted by the mere size of the structures being compared.

4 WDK for sequence labeling and applications to NER

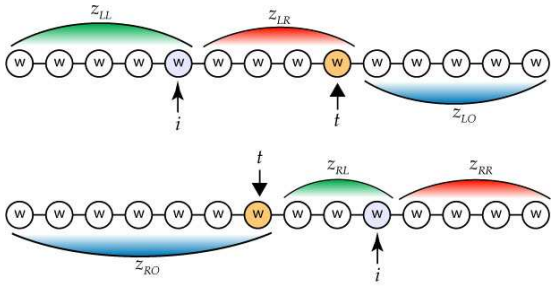
In a sequence labeling task we want to map input sequences to output sequences, or, more precisely, we want to map each element of an input sequence that takes label from a source alphabet to an element with label in a destination alphabet.

Here we cast the sequence labeling task into position specific classification, where different sequence positions give independent examples. This is different from previous approaches in the literature where the sequence labeling problem is solved by searching in the output space (Tsochantaridis et al., 2004; Daumé III and Marcu, 2005). Although the method lacks the potential for collectively labeling all positions simultaneously, it results in a much more efficient algorithm.

In the remainder of the section we introduce a specialized version of the weighted decomposition kernel suitable for a sequence transduction task originating in the natural language processing domain: the named entity recognition (NER) problem, where we map sentences to sequences of a reduced number of named entities (see Sec.4.1).

More formally, given a finite dictionary Σ of words and an input sentence $x \in \Sigma^*$, our input objects are pairs of sentences and indices $r = (x, t)$

Figure 1: Sentence decomposition.



where $r \in \Sigma^* \times \mathbb{N}$. Given a sentence x, two integers $b \geq 1$ and $b \leq e \leq |x|$, let $x[b]$ denote the word at position b and $x[b..e]$ the sub-sequence of x spanning positions from b to e. Finally we will denote by $\xi(x[b])$ a word attribute such as a morphological trait (*is a number* or *has capital initial*, see 4.1) for the word in sentence x at position b.

We introduce two versions of WDK: one with four context types ($D = 4$) and one with increased contextual information ($D = 6$) (see Eq. 5). The relation R depends on two integers t and $i \in \{1, \ldots, |x|\}$, where t indicates the position of the word we want to classify and i the position of a generic word in the sentence. The relation for the first kernel version is defined as: $R = \{(s, z_{LL}, z_{LR}, z_{RL}, z_{RR}, r)\}$ such that the selector $s = x[i]$ is the word at position i, the z_{LL} (LeftLeft) part is a sequence defined as $x[1..i]$ if $i < t$ or the null sequence ε otherwise and the z_{LR} (LeftRight) part is the sequence $x[i + 1..t]$ if $i < t$ or ε otherwise. Informally, z_{LL} is the initial portion of the sentence up to word of position i, and z_{LR} is the portion of the sentence from word at position $i + 1$ up to t (see Fig. 1). Note that when we are dealing with a word that lies to the left of the target word t, its z_{RL} and z_{RR} parts are empty. Symmetrical definitions hold for z_{RL} and z_{RR} when $i > t$. We define the weighted decomposition kernel for sequences as

$$K(r, r') = \sum_{t=1}^{|x|} \sum_{t'=1}^{|x'|} \delta_\xi(s, s') \sum_{d \in \{LL,LR,RL,RR\}} \kappa(z_d, z'_d) \quad (6)$$

where $\delta_\xi(s, s') = 1$ if $\xi(s) = \xi(s')$ and 0 otherwise (that is δ_ξ checks whether the two selector words have the same morphological trait) and κ is Eq. 2 with only one attribute which then boils down to Eq. 3 or Eq. 4, that is a kernel over the histogram for word occurrences over a specific part.

Intuitively, when applied to word sequences, this kernel considers separately words to the left

of the entry we want to transduce and those to its right. The kernel computes the similarity for each sub-sequence by matching the corresponding bag of enriched words: each word is matched only with words that have the same trait as extracted by ξ and the match is then weighted proportionally to the frequency count of identical words preceding and following it.

The kernel version with D=6 adds two parts called z_{LO} (LeftOther) and z_{RO} (RightOther) defined as $x[t+1..|r|]$ and $x[1..t]$ respectively; these represent the remaining sequence parts so that $x = z_{LL} \circ z_{LR} \circ z_{LO}$ and $x = z_{RL} \circ z_{RR} \circ z_{RO}$.

Note that the WDK transforms the sentence in a bag of enriched words computed in a preprocessing phase thus achieving a significant reduction in computational complexity (compared to the recursive procedure in (Lodhi et al., 2002)).

4.1 Named Entity Recognition Experimental Results

Named entities are phrases that contain the names of persons, organizations, locations, times and quantities. For example in the following sentence:

```
[PER Wolff ] , currently a journalist in [LOC
Argentina ] , played with [PER Del Bosque ] in the
final years of the seventies in [ORG Real Madrid].
```

we are interested in predicting that Wolff and Del Bosque are people's names, that Argentina is a name of a location and that Real Madrid is a name of an organization.

The chosen dataset is provided by the shared task of CoNLL–2002 (Saunders et al., 2002) which concerns language–independent named entity recognition. There are four types of phrases: person names (PER), organizations (ORG), locations (LOC) and miscellaneous names (MISC), combined with two tags, B to denote the first item of a phrase and I for any non–initial word; all other phrases are classified as (OTHER). Of the two available languages (Spanish and Dutch), we run experiments only on the Spanish dataset which is a collection of news wire articles made available by the Spanish EFE News Agency. We select a subset of 300 sentences for training and we evaluate the performance on test set. For each category, we evaluate the $F_{\beta=1}$ measure of 4 versions of WDK: word histograms are matched using HIK (Eq. 3) and the kernels on various parts (z_{LL}, z_{LR},etc) are combined with a summation SUMHIK or product PROHIK; alternatively the histograms are combined with a PPK (Eq. 4) obtaining SUMPPK, PROPPK.

Table 1: NER experiment D=4

CLASS	SUMHIS	PROHIS	SUMPRO	PROPRO
B-LOC	74.33	68.68	72.12	66.47
I-LOC	58.18	52.76	59.24	52.62
B-MISC	52.77	43.31	46.86	39.00
I-MISC	79.98	80.15	77.85	79.65
B-ORG	69.00	66.87	68.42	67.52
I-ORG	76.25	75.30	75.12	74.76
B-PER	60.11	56.60	59.33	54.80
I-PER	65.71	63.39	65.67	60.98
MICRO $F_{\beta=1}$	69.28	66.33	68.03	65.30

Table 2: NER experiment with D=6

CLASS	SUMHIS	PROHIS	SUMPRO	PROPRO
B-LOC	74.81	73.30	73.65	73.69
I-LOC	57.28	58.87	57.76	59.44
B-MISC	56.54	64.11	57.72	62.11
I-MISC	78.74	84.23	79.27	83.04
B-ORG	70.80	73.02	70.48	73.10
I-ORG	76.17	78.70	74.26	77.51
B-PER	66.25	66.84	66.04	67.46
I-PER	68.06	71.81	69.55	69.55
MICRO $F_{\beta=1}$	70.69	72.90	70.32	72.38

The word attribute considered for the selector is a word morphologic trait that classifies a word in one of five possible categories: normal word, number, all capital letters, only capital initial and contains non alphabetic characters, while the context histograms are computed counting the exact word frequencies.

Results reported in Tab. 1 and Tab. 2 show that performance is mildly affected by the different choices on how to combine information on the various contexts, though it seems clear that increasing contextual information has a positive influence.

Note that interesting preliminary results can be obtained even without the use of any refined language knowledge, such as part of speech tagging or shallow/deep parsing.

5 Kernels for word semantic ambiguity

Parsing a natural language sentence often involves the choice between different syntax structures that are equally admissible in the given grammar. One of the most studied ambiguity arise when deciding between attaching a prepositional phrase either to the noun phrase or to the verb phrase. An example could be:

1. *eat salad with forks* (attach to verb)
2. *eat salad with tomatoes* (attach to noun)

The resolution of such ambiguities is usually performed by the human reader using its past experiences and the knowledge of the words meaning. Machine learning can simulate human experience by using corpora of disambiguated phrases to compute a decision on new cases. However, given the number of different words that are currently used in texts, there would never be a sufficient dataset from which to learn. Adding semantic information on the possible word meanings would permit the learning of rules that apply to entire categories and can be generalized to all the member words.

5.1 Adding Semantic with WordNet

WordNet (Fellbaum, 1998) is an electronic lexical database of English words built and annotated by linguistic researchers. WordNet is an extensive and reliable source of semantic information that can be used to enrich the representation of a word. Each word is represented in the database by a group of *synonym sets* (synset), with each synset corresponding to an individual linguistic concept. All the synsets contained in WordNet are linked by relations of various types. An important relation connects a synset to its *hypernyms*, that are its immediately broader concepts. The hypernym (and its opposite *hyponym*) relation defines a semantic hierarchy of synsets that can be represented as a directed acyclic graph. The different lexical categories (verbs, nouns, adjectives and adverbs) are contained in distinct hierarchies and each one is rooted by many synsets.

Several metrics have been devised to compute a similarity score between two words using WordNet. In the following we resort to a multiset version of the proximity measure used in (Siolas and d'Alche Buc, 2000), though more refined alternatives are also possible (for example using the conceptual density as in (Basili et al., 2005)). Given the acyclic nature of the semantic hierarchies, each synset can be represented by a group of paths that follows the hypernym relations and finish in one of the top level concepts. Two paths can then be compared by counting how many steps from the roots they have in common. This number must then be normalized dividing by the square root of the product between the path lengths. In this way one can accounts for the unbalancing that arise from different parts of the hierarchies being differently detailed. Given two paths π and π', let l and l' be their lengths and n be the size of their common part, the resulting kernel is:

$$k(\pi, \pi') = \frac{n}{\sqrt{l \cdot l'}} \tag{7}$$

The demonstration that k is positive definite arise from the fact that n can be computed as a positive kernel k^* by summing the exact match kernels between the corresponding positions in π and π' seen as sequences of synset identifiers. The lengths l and l' can then be evaluated as $k^*(\pi, \pi)$ and $k^*(\pi', \pi')$ and k is the resulting normalized version of k^*.

The kernel between two synsets σ and σ' can then be computed by the multi-set kernel (Gärtner et al., 2002a) between their corresponding paths. Synsets are organized into forty-five lexicographer files based on syntactic category and logical groupings. Additional information can be derived by comparing the identifiers λ and λ' of these file associated to σ and σ'. The resulting synset kernel is:

$$\kappa_\sigma(\sigma, \sigma') = \delta(\lambda, \lambda') + \sum_{\pi \in \Pi} \sum_{\pi' \in \Pi'} k(\pi, \pi') \tag{8}$$

where Π is the set of paths originating from σ and the exact match kernel $\delta(\lambda, \lambda')$ is 1 if $\lambda \equiv \lambda'$ and 0 otherwise. Finally, the kernel κ_ω between two words is itself a multi-set kernel between the corresponding sets of synsets:

$$\kappa_\omega(\omega, \omega') = \sum_{\sigma \in \Sigma} \sum_{\sigma' \in \Sigma'} \kappa_\sigma(\sigma, \sigma') \tag{9}$$

where Σ are the synsets associated to the word ω.

5.2 PP Attachment Experimental Results

The experiments have been performed using the Wall-Street Journal dataset described in (Ratnaparkhi et al., 1994). This dataset contains $20,800$ training examples and $3,097$ testing examples. Each phrase x in the dataset is reduced to a verb x_v, its object noun x_{n_1} and prepositional phrase formed by a preposition x_p and a noun x_{n_2}. The target is either V or N whether the phrase is attached to the verb or the noun. Data have been preprocessed by assigning to all the words their corresponding synsets. Additional meanings derived from specific synsets have been attached to the words as described in (Stetina and Nagao, 1997). The kernel between two phrases x and x' is then computed by combining the kernels between single words using either the sum or the product.

Method	Acc	Pre	Rec
S	$84.6\% \pm 0.65\%$	90.8%	82.2%
P	$84.8\% \pm 0.65\%$	92.2%	81.0%
SW	$85.4\% \pm 0.64\%$	90.9%	83.6%
SWL	$85.3\% \pm 0.64\%$	91.1%	83.2%
PW	$85.9\% \pm 0.63\%$	92.2%	83.1%
PWL	$86.2\% \pm 0.62\%$	92.1%	83.7%

Table 3: Summary of the experimental results on the PP attachment problem for various kernel parameters.

Results of the experiments are reported in Tab. 3 for various kernels parameters: S or P denote if the sum or product of the kernels between words are used, W denotes that WordNet semantic information is added (otherwise the kernel between two words is just the exact match kernel) and L denotes that lexicographer files identifiers are used. An additional gaussian kernel is used on top of K_{pp}. The C and γ parameters are selected using an independent validation set. For each setting, accuracy, precision and recall values on the test data are reported, along with the standard deviation of the estimated binomial distribution of errors. The results demonstrate that semantic information can help in resolving PP ambiguities. A small difference exists between taking the product instead of the sum of word kernels, and an additional increase in the amount of information available to the learner is given by the use of lexicographer files identifiers.

6 Using declarative knowledge for NLP kernel integration

Data objects in NLP often require complex representations; suffice it to say that a sentence is naturally represented as a variable length sequence of word tokens and that shallow/deep parsers are reliably used to enrich these representations with links between words to form parse trees. Finally, additional complexity can be introduced by including semantic information. Various facets of this richness of representations have been extensively investigated, including the expressiveness of various grammar formalisms, the exploitation of lexical representation (e.g. verb subcategorization, semantic tagging), and the use of machine readable sources of generic or specialized knowledge (dictionaries, thesauri, domain specific ontologies). All these efforts are capable to address language specific sub-problems but their integration into a coherent framework is a difficult feat.

Recent ideas for constructing kernel functions starting from logical representations may offer an appealing solution. Gärtner et al. (2002) have proposed a framework for defining kernels on higher-order logic individuals. Cumby and Roth (2003) used description logics to represent knowledge jointly with propositionalization for defining a kernel function. Frasconi et al. (2004) proposed kernels for handling supervised learning in a setting similar to that of inductive logic programming where data is represented as a collection of facts and background knowledge by a declarative program in first-order logic. In this section, we briefly review this approach and suggest a possible way of exploiting it for the integration of different sources of knowledge that may be available in NLP.

6.1 Declarative Kernels

The definition of decomposition kernels as reported in Section 2 is very general and covers almost all kernels for discrete structured data developed in the literature so far. Different kernels are designed by defining the relation decomposing an example into its "parts", and specifying kernels for individual parts. In (Frasconi et al., 2004) we proposed a systematic approach to such design, consisting in formally defining a relation by the set of axioms it must satisfy. We relied on *mereotopology* (Varzi, 1996) (i.e. the theory of parts and places) in order to give a formal definition of the intuitive concepts of parthood and connection. The formalization of mereotopological relations allows to automatically deduce instances of such relations on the data, by exploiting the background knowledge which is typically available on structured domains. In (Frasconi et al., 2004) we introduced *declarative kernels* (DK) as a set of kernels working on mereotopological relations, such as that of proper parthood ($\prec_P$) or more complex relations based on connected parts. A typed syntax for objects was introduced in order to provide additional flexibility in defining kernels on instances of the given relation. A basic kernel on parts K_P was defined as follows:

$$K_P(x,x') = \sum_{\substack{s \prec_P x \\ s' \prec_P x'}} \delta_T(s,s') \big(\kappa(s,s') + K_P(s,s')\big) \quad (10)$$

where δ_T matches objects of the same type and κ is a kernel over object attributes.

Declarative kernels were tested in (Frasconi et al., 2004) on a number of domains with promising results, including a biomedical information extraction task (Goadrich et al., 2004) aimed at detecting protein-localization relationships within Medline abstracts. A DK on parts as the one defined in Eq. (10) outperformed state-of-the-art ILP-based systems Aleph and Gleaner (Goadrich et al., 2004) in such information extraction task, and required about three orders of magnitude less training time.

6.2 Weighted Decomposition Declarative Kernels

Declarative kernels can be combined with WDK in a rather straightforward way, thus taking the advantages of both methods. A simple approach is that of using proper parthood in place of selectors, and topology to recover the context of each proper part. A weighted decomposition declarative kernel (WD^{2}K) of this kind would be defined as in Eq. (10) simply adding to the attribute kernel κ a context kernel that compares the surrounding of a pair of objects—as defined by the topology relation—using some aggregate kernel such as PPK or HIK (see Section 3). Note that such definition extends WDK by adding recursion to the concept of comparison by selector, and DK by adding contexts to the kernel between parts. Multiple contexts can be easily introduced by employing different notions of topology, provided they are consistent with mereotopological axioms. As an example, if objects are words in a textual document, we can define l-connection as the relation for which two words are l-connected if there are consequential within the text with at most l words in between, and obtain growingly large contexts by increasing l. Moreover, an extended representation of words, as the one employing WordNet semantic information, could be easily plugged in by including a kernel for synsets such as that in Section 5.1 into the kernel κ on word attributes. Additional relations could be easily formalized in order to exploit specific linguisitc knowledge: a causal relation would allow to distinguish between preceding and following context so to take into consideration word order; an underlap relation, associating two objects being parts of the same super-object (i.e. pre-terminals dominated by the same non-terminal node), would be able to express commanding notions.

The promising results obtained with declarative kernels (where only very simple lexical information was used) together with the declarative ease to integrate arbitrary kernels on specific parts are all encouraging signs that boost our confidence in this line of research.

References

Roberto Basili, Marco Cammisa, and Alessandro Moschitti. 2005. Effective use of wordnet semantics via kernel-based learning. In *9th Conference on Computational Natural Language Learning*, Ann Arbor(MI), USA.

S. Ben-David, N. Eiron, and H. U. Simon. 2002. Limitations of learning via embeddings in euclidean half spaces. *J. of Mach. Learning Research*, 3:441–461.

M. Collins and N. Duffy. 2002. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *Proceedings of the Fortieth Annual Meeting on Association for Computational Linguistics*, pages 263–270, Philadelphia, PA, USA.

C. Cortes, P. Haffner, and M. Mohri. 2004. Rational kernels: Theory and algorithms. *J. of Machine Learning Research*, 5:1035–1062.

A. Culotta and J. Sorensen. 2004. Dependency tree kernels for relation extraction. In *Proc. of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 423–429.

C. M. Cumby and D. Roth. 2003. On kernel methods for relational learning. In *Proc. Int. Conference on Machine Learning (ICML'03)*, pages 107–114, Washington, DC, USA.

H. Daumé III and D. Marcu. 2005. Learning as search optimization: Approximate large margin methods for structured prediction. In *International Conference on Machine Learning (ICML)*, pages 169–176, Bonn, Germany.

C. Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. The MIT Press.

P. Frasconi, S. Muggleton, H. Lodhi, and A. Passerini. 2004. Declarative kernels. Technical Report RT 2/2004, Università di Firenze.

T. Gärtner, P. A. Flach, A. Kowalczyk, and A. J. Smola. 2002a. Multi-instance kernels. In C. Sammut and A. Hoffmann, editors, *Proceedings of the 19th International Conference on Machine Learning*, pages 179–186. Morgan Kaufmann.

T. Gärtner, J.W. Lloyd, and P.A. Flach. 2002b. Kernels for structured data. In S. Matwin and C. Sammut, editors, *Proceedings of the 12th International Conference on Inductive Logic Programming*, volume 2583 of *Lecture Notes in Artificial Intelligence*, pages 66–83. Springer-Verlag.

T. Gärtner. 2003. A survey of kernels for structured data. *SIGKDD Explorations Newsletter*, 5(1):49–58.

M. Goadrich, L. Oliphant, and J. W. Shavlik. 2004. Learning ensembles of first-order clauses for recall-precision curves: A case study in biomedical information extraction. In *Proc. 14th Int. Conf. on Inductive Logic Programming, ILP '04*, pages 98–115.

D. Haussler. 1999. Convolution kernels on discrete structures. Technical Report UCSC-CRL-99-10, University of California, Santa Cruz.

T. Horváth, T. Gärtner, and S. Wrobel. 2004. Cyclic pattern kernels for predictive graph mining. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 158–167. ACM Press.

T. Jebara, R. Kondor, and A. Howard. 2004. Probability product kernels. *J. Mach. Learn. Res.*, 5:819–844.

H. Kashima and T. Koyanagi. 2002. Kernels for Semi–Structured Data. In *Proceedings of the Nineteenth International Conference on Machine Learning*, pages 291–298.

H. Kashima, K. Tsuda, and A. Inokuchi. 2003. Marginalized kernels between labeled graphs. In *Proceedings of the Twentieth International Conference on Machine Learning*, pages 321–328, Washington, DC, USA.

C. S. Leslie, E. Eskin, and W. S. Noble. 2002. The spectrum kernel: A string kernel for SVM protein classification. In *Pacific Symposium on Biocomputing*, pages 566–575.

H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins. 2002. Text classification using string kernels. *Journal of Machine Learning Research*, 2:419–444.

S. Menchetti, F. Costa, and P. Frasconi. 2005. Weighted decomposition kernels. In *Proceedings of the Twenty-second International Conference on Machine Learning*, pages 585–592, Bonn, Germany.

Alessandro Moschitti. 2004. A study on convolution kernels for shallow semantic parsing. In *42-th Conference on Association for Computational Linguistic*, Barcelona, Spain.

F. Odone, A. Barla, and A. Verri. 2005. Building kernels from binary strings for image matching. *IEEE Transactions on Image Processing*, 14(2):169–180.

A Ratnaparkhi, J. Reynar, and S. Roukos. 1994. A maximum entropy model for prepositional phrase attachment. In *Proceedings of the ARPA Human Language Technology Workshop*, pages 250–255, Plainsboro, NJ.

C. Saunders, H. Tschach, and J. Shawe-Taylor. 2002. Syllables and other string kernel extensions. In *Proceedings of the Nineteenth International Conference on Machine Learning*, pages 530–537.

B. Schölkopf, J. Weston, E. Eskin, C. S. Leslie, and W. S. Noble. 2002. A kernel approach for learning from almost orthogonal patterns. In *Proc. of ECML'02*, pages 511–528.

J. Shawe-Taylor and N. Cristianini. 2004. *Kernel Methods for Pattern Analysis*. Cambridge University Press.

G. Siolas and F. d'Alche Buc. 2000. Support vector machines based on a semantic kernel for text categorization. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks*, volume 5, pages 205 – 209.

A.J. Smola and R. Kondor. 2003. Kernels and regularization on graphs. In B. Schölkopf and M.K. Warmuth, editors, *16th Annual Conference on Computational Learning Theory and 7th Kernel Workshop, COLT/Kernel 2003*, volume 2777 of *Lecture Notes in Computer Science*, pages 144–158. Springer.

J Stetina and M Nagao. 1997. Corpus based pp attachment ambiguity resolution with a semantic dictionary. In *Proceedings of the Fifth Workshop on Very Large Corpora*, pages 66–80, Beijing, China.

J. Suzuki, H. Isozaki, and E. Maeda. 2004. Convolution kernels with feature selection for natural language processing tasks. In *Proc. of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 119–126.

I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. 2004. Support vector machine learning for interdependent and structured output spaces. In *Proc. 21st Int. Conf. on Machine Learning*, pages 823–830, Banff, Alberta, Canada.

A.C. Varzi. 1996. Parts, wholes, and part-whole relations: the prospects of mereotopology. *Data and Knowledge Engineering*, 20:259–286.

D. Zelenko, C. Aone, and A. Richardella. 2003. Kernel methods for relation extraction. *Journal of Machine Learning Research*, 3:1083–1106.

A Multiclassifier based Document Categorization System: profiting from the Singular Value Decomposition Dimensionality Reduction Technique

Ana Zelaia
UPV-EHU
Basque Country
ccpzejaa@si.ehu.es

Iñaki Alegria
UPV-EHU
Basque Country
acpalloi@si.ehu.es

Olatz Arregi
UPV-EHU
Basque Country
acparuro@si.ehu.es

Basilio Sierra
UPV-EHU
Basque Country
ccpsiarb@si.ehu.es

Abstract

In this paper we present a multiclassifier approach for multilabel document classification problems, where a set of k-NN classifiers is used to predict the category of text documents based on different training subsampling databases. These databases are obtained from the original training database by random subsampling. In order to combine the predictions generated by the multiclassifier, Bayesian voting is applied. Through all the classification process, a reduced dimension vector representation obtained by Singular Value Decomposition (SVD) is used for training and testing documents. The good results of our experiments give an indication of the potentiality of the proposed approach.

1 Introduction

Document Categorization, the assignment of natural language texts to one or more predefined categories based on their content, is an important component in many information organization and management tasks. Researchers have concentrated their efforts in finding the appropriate way to represent documents, index them and construct classifiers to assign the correct categories to each document. Both, document representation and classification method are crucial steps in the categorization process.

In this paper we concentrate on both issues. On the one hand, we use Latent Semantic Indexing (LSI) (Deerwester et al., 1990), which is a variant of the vector space model (VSM) (Salton and McGill, 1983), in order to obtain the vector representation of documents. This technique compresses vectors representing documents into vectors of a lower-dimensional space. LSI, which is based on Singular Value Decomposition (SVD) of matrices, has showed to have the ability to extract the relations among words and documents by means of their context of use, and has been successfully applied to Information Retrieval tasks.

On the other hand, we construct a multiclassifier (Ho et al., 1994) which uses different training databases. These databases are obtained from the original training set by random subsampling. We implement this approach by bagging, and use the k-NN classification algorithm to make the category predictions for testing documents. Finally, we combine all predictions made for a given document by Bayesian voting.

The experiment we present has been evaluated for Reuters-21578 standard document collection. Reuters-21578 is a multilabel document collection, which means that categories are not mutually exclusive because the same document may be relevant to more than one category. Being aware of the results published in the most recent literature, and having obtained good results in our experiments, we consider the categorization method presented in this paper an interesting contribution for text categorization tasks.

The remainder of this paper is organized as follows: Section 2, discusses related work on document categorization for Reuters-21578 collection. In Section 3, we present our approach to deal with the multilabel text categorization task. In Section 4 the experimental setup is introduced, and details about the Reuters database, the preprocessing applied and some parameter setting are provided. In Section 5, experimental results are presented and discussed. Finally, Section 6 contains some conclusions and comments on future work.

2 Related Work

As previously mentioned in the introduction, text categorization consists in assigning predefined categories to text documents. In the past two decades, document categorization has received much attention and a considerable number of machine learning based approaches have been proposed. A good tutorial on the state-of-the-art of document categorization techniques can be found in (Sebastiani, 2002).

In the document categorization task we can find two cases; (1) the multilabel case, which means that categories are not mutually exclusive, because the same document may be relevant to more than one category (1 to m category labels may be assigned to the same document, being m the total number of predefined categories), and (2) the single-label case, where exactly one category is assigned to each document. While most machine learning systems are designated to handle multiclass data[1], much less common are systems that can handle multilabel data.

For experimentation purposes, there are standard document collections available in the public domain that can be used for document categorization. The most widely used is Reuters-21578 collection, which is a multiclass (135 categories) and multilabel (the mean number of categories assigned to a document is 1.2) dataset. Many experiments have been carried out for the Reuters collection. However, they have been performed in different experimental conditions. This makes results difficult to compare among them. In fact, effectiveness results can only be compared between studies that use the same training and testing sets. In order to lead researchers to use the same training/testing divisions, the Reuters documents have been specifically tagged, and researchers are encouraged to use one of those divisions. In our experiment we use the "ModApte" split (Lewis, 2004).

In this section, we analize the category subsets, evaluation measures and results obtained in the past and in the recent years for Reuters-21578 ModApte split.

2.1 Category subsets

Concerning the evaluation of the classification system, we restrict our attention to the TOPICS group of categories that labels Reuters dataset, which contains 135 categories. However, many categories appear in no document and consequently, and because inductive based learning classifiers learn from training examples, these categories are not usually considered at evaluation time. The most widely used subsets are the following:

- Top-10: It is the set of the 10 categories which have the highest number of documents in the training set.
- R(90): It is the set of 90 categories which have at least one document in the training set and one in the testing set.
- R(115): It is the set of 115 categories which have at least one document in the training set.

In order to analyze the relative hardness of the three category subsets, a very recent paper has been published by Debole and Sebastiani (Debole and Sebastiani, 2005) where a systematic, comparative experimental study has been carried out.

The results of the classification system we propose are evaluated according to these three category subsets.

2.2 Evaluation measures

The evaluation of a text categorization system is usually done experimentally, by measuring the effectiveness, i.e. average correctness of the categorization. In binary text categorization, two known statistics are widely used to measure this effectiveness: precision and recall. Precision (Prec) is the percentage of documents correctly classified into a given category, and recall (Rec) is the percentage of documents belonging to a given category that are indeed classified into it.

In general, there is a trade-off between precision and recall. Thus, a classifier is usually evaluated by means of a measure which combines precision and recall. Various such measures have been proposed. The breakeven point, the value at which precision equals recall, has been frequently used during the past decade. However, it has been recently criticized by its proposer ((Sebastiani, 2002) footnote 19). Nowadays, the F_1 score is more frequently used. The F_1 score combines recall and precision with an equal weight in the following way:

$$F_1 = \frac{2 \cdot \text{Prec} \cdot \text{Rec}}{\text{Prec} + \text{Rec}}$$

[1] Categorization problems where there are more than two possible categories.

Since precision and recall are defined only for binary classification tasks, for multiclass problems results need to be averaged to get a single performance value. This will be done using *microaveraging* and *macroaveraging*. In microaveraging, which is calculated by globally summing over all individual cases, categories count proportionally to the number of their positive testing examples. In macroaveraging, which is calculated by averaging over the results of the different categories, all categories count the same. See (Debole and Sebastiani, 2005; Yang, 1999) for more detailed explanation of the evaluation measures mentioned above.

2.3 Comparative Results

Sebastiani (Sebastiani, 2002) presents a table where lists results of experiments for various training/testing divisions of Reuters. Although we are aware that the results listed are microaveraged breakeven point measures, and consequently, are not directly comparable to the ones we present in this paper, F_1, we want to remark some of them. In Table 1 we summarize the best results reported for the ModApte split listed by Sebastiani.

Results reported by	R(90)	Top-10
(Joachims, 1998)	86.4	
(Dumais et al., 1998)	87.0	92.0
(Weiss et.al., 1999)	87.8	

Table 1: Microaveraged breakeven point results reported by Sebastiani for the Reuters-21578 ModApte split.

In Table 2 we include some more recent results, evaluated according to the microaveraged F_1 score. For R(115) there is also a good result, $F_1 = 87.2$, obtained by (Zhang and Oles, 2001)[2].

3 Proposed Approach

In this paper we propose a multiclassifier based document categorization system. Documents in the training and testing sets are represented in a reduced dimensional vector space. Different training databases are generated from the original training dataset in order to construct the multiclassifier. We use the k-NN classification algorithm, which according to each training database makes a prediction for testing documents. Finally, a Bayesian voting scheme is used in order to definitively assign category labels to testing documents.

[2]Actually, this result is obtained for 118 categories which correspond to the 115 mentioned before and three more categories which have testing documents but no training document assigned.

Results reported by	R(90)	Top-10
(Gao et al., 2003)	88.42	93.07
(Kim et al., 2005)	87.11	92.21
(Gliozzo and Strapparava, 2005)		92.80

Table 2: F_1 results reported for the Reuters-21578 ModApte split.

In the rest of this section we make a brief review of the SVD dimensionality reduction technique, the k-NN algorithm and the combination of classifiers used.

3.1 The SVD Dimensionality Reduction Technique

The classical Vector Space Model (VSM) has been successfully employed to represent documents in text categorization tasks. The newer method of Latent Semantic Indexing (LSI) [3] (Deerwester et al., 1990) is a variant of the VSM in which documents are represented in a lower dimensional space created from the input training dataset. It is based on the assumption that there is some underlying latent semantic structure in the term-document matrix that is corrupted by the wide variety of words used in documents. This is referred to as the problem of polysemy and synonymy. The basic idea is that if two document vectors represent two very similar topics, many words will co-occur on them, and they will have very close semantic structures after dimension reduction.

The SVD technique used by LSI consists in factoring term-document matrix M into the product of three matrices, $M = U\Sigma V^T$ where Σ is a diagonal matrix of singular values in non-increasing order, and U and V are orthogonal matrices of singular vectors (term and document vectors, respectively). Matrix M can be approximated by a lower rank M_p which is calculated by using the p largest singular values of M. This operation is called dimensionality reduction, and the p-dimensional

[3]http://lsi.research.telcordia.com, http://www.cs.utk.edu/∼lsi

space to which document vectors are projected is called the reduced space. Choosing the right dimension p is required for successful application of the LSI/SVD technique. However, since there is no theoretical optimum value for it, potentially expensive experimentation may be required to determine it (Berry and Browne, 1999).

For document categorization purposes (Dumais, 2004), the testing document q is also projected to the p-dimensional space, $q_p = q^T U_p \Sigma_p^{-1}$, and the cosine is usually calculated to measure the semantic similarity between training and testing document vectors.

In Figure 1 we can see an ilustration of the document vector projection. Documents in the training collection are represented by using the term-document matrix M, and each one of the documents is represented by a vector in the $\mathbb{R}^m$ vector space like in the traditional vector space model (VSM) scheme. Afterwards, the dimension p is selected, and by applying SVD vectors are projected to the reduced space. Documents in the testing collection will also be projected to the same reduced space.

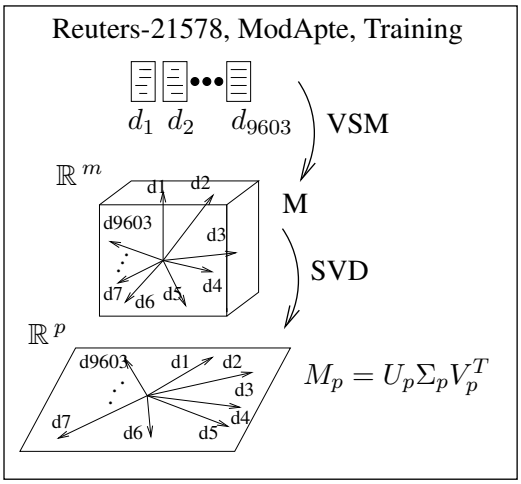


Figure 1: Vectors in the VSM are projected to the reduced space by using SVD.

3.2 The k nearest neighbor classification algorithm (k-NN)

k-NN is a distance based classification approach. According to this approach, given an arbitrary testing document, the k-NN classifier ranks its nearest neighbors among the training documents, and uses the categories of the k top-ranking neighbors to predict the categories of the testing document (Dasarathy, 1991). In this paper, the training and testing documents are represented as reduced dimensional vectors in the lower dimensional space, and in order to find the nearest neighbors of a given document, we calculate the cosine similarity measure.

In Figure 2 an ilustration of this phase can be seen, where some training documents and a testing document q are projected in the $\mathbb{R}^p$ reduced space. The nearest to the q_p testing document are considered to be the vectors which have the smallest angle with q_p. According to the category labels of the nearest documents, a category label prediction, c, will be made for testing document q.

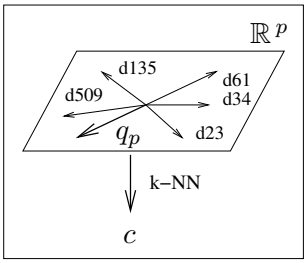


Figure 2: The k-NN classifier is applied to q_p testing document and c category label is predicted.

We have decided to use the k-NN classifier because it has been found that on the Reuters-21578 database it performs best among the conventional methods (Joachims, 1998; Yang, 1999) and because we have obtained good results in our previous work on text categorization for documents written in Basque, a highly inflected language (Zelaia et al., 2005). Besides, the k-NN classification algorithm can be easily adapted to multilabel categorization problems such as Reuters.

3.3 Combination of classifiers

The combination of multiple classifiers has been intensively studied with the aim of improving the accuracy of individual components (Ho et al., 1994). Two widely used techniques to implement this approach are *bagging* (Breiman, 1996), that uses more than one model of the same paradigm; and *boosting* (Freund and Schapire, 1999), in which a different weight is given to different training examples looking for a better accuracy.

In our experiment we have decided to construct a multiclassifier via bagging. In bagging, a set of training databases TD_i is generated by selecting n training examples drawn randomly with replacement from the original training database TD of n examples. When a set of n_1 training examples,

$n_1 < n$, is chosen from the original training collection, the bagging is said to be applied by random subsampling. This is the approach used in our work. The n_1 parameter has been selected via tuning. In Section 4.3 the selection will be explained in a more extended way.

According to the random subsampling, given a testing document q, the classifier will make a label prediction c^i based on each one of the training databases TD_i. One way to combine the predictions is by Bayesian voting (Dietterich, 1998), where a confidence value $\text{cv}^i_{c_j}$ is calculated for each training database TD_i and category c_j to be predicted. These confidence values have been calculated based on the original training collection. Confidence values are summed by category. The category c_j that gets the highest value is finally proposed as a prediction for the testing document.

In Figure 3 an ilustration of the whole experiment can be seen. First, vectors in the VSM are projected to the reduced space by using SVD. Next, random subsampling is applied to the training database TD to obtain different training databases TD_i. Afterwards the k-NN classifier is applied for each TD_i to make category label predictions. Finally, Bayesian voting is used to combine predictions, and c_j, and in some cases c_k as well, will be the final category label prediction of the categorization system for testing document q. In Section 4.3 the cases when a second category label prediction c_k is given are explained.

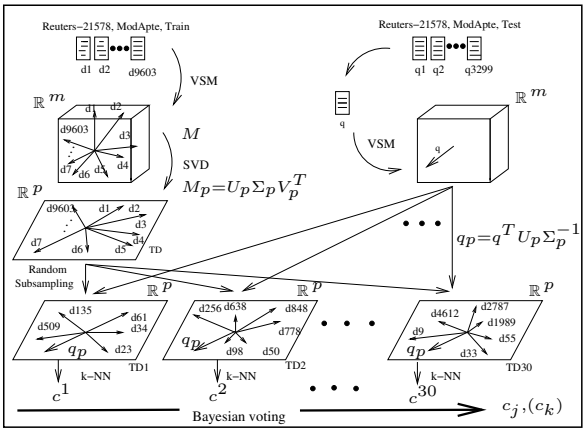


Figure 3: Proposed approach for multilabel document categorization tasks.

4 Experimental Setup

The aim of this section is to describe the document collection used in our experiment and to give an account of the preprocessing techniques and parameter settings we have applied.

When machine learning and other approaches are applied to text categorization problems, a common technique has been to decompose the multiclass problem into multiple, independent binary classification problems. In this paper, we adopt a different approach. We will be primarily interested in a classifier which produces a ranking of possible labels for a given document, with the hope that the appropriate labels will appear at the top of the ranking.

4.1 Document Collection

As previously mentioned, the experiment reported in this paper has been carried out for the Reuters-21578 dataset[4] compiled by David Lewis and originally collected by the Carnegie group from the Reuters newswire in 1987. We use one of the most widely used training/testing divisions, the "ModApte" split, in which 75 % of the documents (9,603 documents) are selected for training and the remaining 25 % (3299 documents) to test the accuracy of the classifier.

Document distribution over categories in both the training and the testing sets is very unbalanced: the 10 most frequent categories, top-10, account 75% of the training documents; the rest is distributed among the other 108 categories.

According to the number of labels assigned to each document, many of them (19% in training and 8.48% in testing) are not assigned to any category, and some of them are assigned to 12. We have decided to keep the unlabeled documents in both the training and testing collections, as it is suggested in (Lewis, 2004)[5].

4.2 Preprocessing

The original format of the text documents is in SGML. We perform some preprocessing to filter out the unused parts of a document. We preserved only the title and the body text, punctuation and numbers have been removed and all letters have been converted to lowercase. We have

[4] http://daviddlewis.com/resources/testcollections

[5] In the "ModApte" Split section it is suggested as follows: "If you are using a learning algorithm that requires each training document to have at least TOPICS category, you can screen out the training documents with no TOPICS categories. Please do NOT screen out any of the 3,299 documents - that will make your results incomparable with other studies."

used the tools provided in the web[6] in order to extract text and categories from each document. We have stemmed the training and testing documents by using the Porter stemmer (Porter, 1980)[7]. By using it, case and flection information are removed from words. Consequently, the same experiment has been carried out for the two forms of the document collection: word-forms and Porter stems.

According to the dimension reduction, we have created the matrices for the two mentioned document collection forms. The sizes of the training matrices created are 15591×9603 for word-forms and 11114×9603 for Porter stems. Different number of dimensions have been experimented ($p = 100, 300, 500, 700$).

4.3 Parameter setting

We have designed our experiment in order to optimize the microaveraged F_1 score. Based on previous experiments (Zelaia et al., 2005), we have set parameter k for the k-NN algorithm to $k = 3$. This way, the k-NN classifier will give a category label prediction based on the categories of the 3 nearest ones.

On the other hand, we also needed to decide the number of training databases TD_i to create. It has to be taken into account that a high number of training databases implies an increasing computational cost for the final classification system. We decided to create 30 training databases. However, this is a parameter that has not been optimized.

There are two other parameters which have been tuned: the size of each training database and the threshold for multilabeling. We now briefly give some cues about the tuning performed.

4.3.1 The size of the training databases

As we have previously mentioned, documents have been randomly selected from the original training database in order to construct the 30 training databases TD_i used in our classification system. There are $n = 9,603$ documents in the original Reuters training collection. We had to decide the number of documents to select in order to construct each TD_i. The number of documents selected from each category preserves the proportion of documents in the original one. We have experimented to select different numbers $n_1 < n$ of documents, according to the following formula:

$$n_1 = \sum_{i=1}^{115} 2 + \frac{t_i}{j}, \qquad j = 10, 20, \ldots, 70,$$

where t_i is the total number of training documents in category i. In Figure 4 it can be seen the variation of the n_1 parameter depending on the value of parameter j. We have experimented different j values, and evaluated the results. Based on the results obtained we decided to select $j = 60$, which means that each one of the 30 training databases will have $n_1 = 298$ documents. As we can see, the final classification system will be using training databases which are quite smaller that the original one. This gives a lower computational cost, and makes the classification system faster.

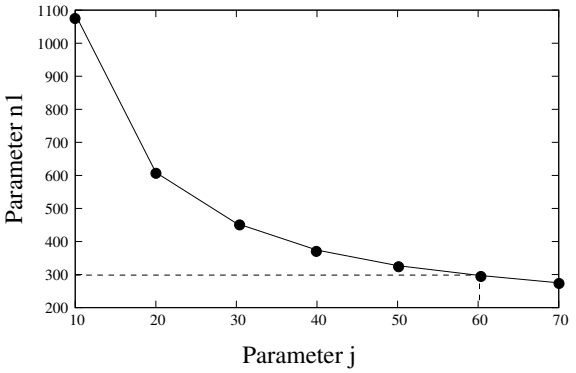


Figure 4: Random subsampling rate.

4.3.2 Threshold for multilabeling

The k-NN algorithm predicts a unique category label for each testing document, based on the ranked list of categories obtained for each training database TD_i[8]. As previously mentioned, we use Bayesian voting to combine the predictions.

The Reuters-21578 is a multilabel database, and therefore, we had to decide in which cases to assign a second category label to a testing document. Given that c_j is the category with the highest value in Bayesian voting and c_k the next one, the second c_k category label will be assigned when the following relation is true:

$$\mathrm{cv}_{c_k} > \mathrm{cv}_{c_j} \times r, \qquad r = 0.1, 0.2, \ldots, 0.9, 1$$

In Figure 5 we can see the mean number of categories assigned to a document for different values

[6] http://www.lins.fju.edu.tw/~tseng/Collections/Reuters-21578.html

[7] http://tartarus.org/martin/PorterStemmer/

[8] It has to be noted that unlabeled documents have been preserved, and thus, our classification system treats unlabeled documents as documents of a new category

of r. Results obtained were evaluated and based on them we decided to select $r = 0.4$, which corresponds to a ratio of 1.05 categories.

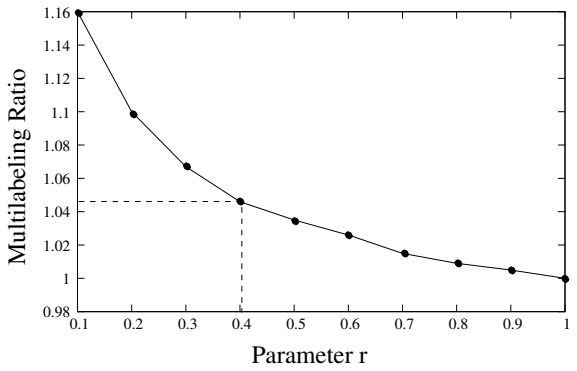


Figure 5: Threshold for multilabeling.

Corpus	Dimension reduction 100	300	500	700
Words(10)	93.06	93.17	93.44	92.00
Porter(10)	**93.57**	93.20	93.50	92.57
Words(90)	84.90	86.71	87.09	86.18
Porter(90)	85.34	86.64	**87.27**	86.30
Words(115)	84.66	86.44	86.73	85.84
Porter(115)	85.13	86.47	**87.01**	86.00

Table 3: Microaveraged F_1 scores for Reuters-21578 ModApte split.

5 Experimental Results

In Table 3 microaveraged F_1 scores obtained in our experiment are shown. As it could be expected, a simple stemming process increases slightly results, and it can be observed that the best result for the three category subsets has been obtained for the stemmed corpus, even though gain is low (less than 0.6).

The evaluation for the Top-10 category subset gives the best results, reaching up to 93.57%. In fact, this is the expected behavior, as the number of categories to be evaluated is small and the number of documents in each category is high. For this subset the best result has been obtained for 100 dimensions, although the variation is low among results for 100, 300 and 500 dimensions. When using higher dimensions results become poorer.

According to the R(90) and R(115) subsets, the best results are 87.27% and 87.01% respectively. Given that the difficulty of these subsets is quite similar, their behavior is also analogous. As we can see in the table, most of the best results for these subsets have been obtained by reducing the dimension of the space to 500.

6 Conclusions and Future Work

In this paper we present an approach for multilabel document categorization problems which consists in a multiclassifier system based on the k-NN algorithm. The documents are represented in a reduced dimensional space calculated by SVD. We want to emphasize that, due to the multilabel character of the database used, we have adapted the classification system in order for it to be multilabel too. The learning of the system has been unique (9603 training documents) and the category label predictions made by the classifier have been evaluated on the testing set according to the three category sets: top-10, R(90) and R(115). The microaveraged F_1 scores we obtain are among the best reported for the Reuters-21578.

As future work, we want to experiment with generating more than 30 training databases, and in a preliminary phase select the best among them. The predictions made using the selected training databases will be combined to obtain the final predictions.

When there is a low number of documents available for a given category, the power of LSI gets limited to create a space that reflects interesting properties of the data. As future work we want to include background text in the training collection and use an expanded term-document matrix that includes, besides the 9603 training documents, some other relevant texts. This may increase results, specially for the categories with less documents (Zelikovitz and Hirsh, 2001).

In order to see the consistency of our classifier, we also plan to repeat the experiment for the RCV1 (Lewis et al., 2004), a new benchmark collection for text categorization tasks which consists of 800,000 manually categorized newswire stories recently made available by Reuters.

7 Acknowledgements

This research was supported by the University of the Basque Country (UPV00141.226-T-15948/2004) and Gipuzkoa Council in a European

Union Program.

References

Berry, M.W. and Browne, M.: Understanding Search Engines: Mathematical Modeling and Text Retrieval. SIAM Society for Industrial and Applied Mathematics, ISBN: 0-89871-437-0, Philadelphia, (1999)

Breiman, L.: Bagging Predictors. Machine Learning, **24**(2), 123–140, (1996)

Cristianini, N., Shawe-Taylor, J. and Lodhi, H.: Latent Semantic Kernels. Proceedings of ICML'01, 18th International Conference on Machine Learning, 66–73, Morgan Kaufmann Publishers, (2001)

Dasarathy, B.V.: Nearest Neighbor (NN) Norms: NN Pattern Recognition Classification Techniques. IEEE Computer Society Press, (1991)

Debole, F. and Sebastiani, F.: An Analysis of the Relative Hardness of Reuters-21578 Subsets. Journal of the American Society for Information Science and Technology, **56**(6),584–596, (2005)

Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K. and Harshman, R.: Indexing by Latent Semantic Analysis. Journal of the American Society for Information Science, **41**, 391–407, (1990)

Dietterich, T.G.: Machine-Learning Research: Four Current Directions. The AI Magazine, **18**(4), 97–136, (1998)

Dumais, S.T., Platt, J., Heckerman, D. and Sahami, M.: Inductive Learning Algorithms and Representations for Text Categorization. Proceedings of CIKM'98: 7th International Conference on Information and Knowledge Management, ACM Press, 148–155 (1998)

Dumais, S.: Latent Semantic Analysis. ARIST, Annual Review of Information Science Technology, **38**, 189–230, (2004)

Freund, Y. and Schapire, R.E.: A Short Introduction to Boosting. Journal of Japanese Society for Artificial Intelligence, **14**(5), 771-780, (1999)

Gao, S., Wu, W., Lee, C.H. and Chua, T.S.: A Maximal Figure-of-Merit Learning Approach to Text Categorization. Proceedings of SIGIR'03: 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 174–181, ACM Press, (2003)

Gliozzo, A. and Strapparava, C.: Domain Kernels for Text Categorization. Proceedings of CoNLL'05: 9th Conference on Computational Natural Language Learning, 56–63, (2005)

Ho, T.K., Hull, J.J. and Srihari, S.N.: Decision Combination in Multiple Classifier Systems. IEEE Transactions on Pattern Analysis and Machine Intelligence, **16**(1), 66–75, (1994)

Joachims, T. Text Categorization with Support Vector Machines: Learning with Many Relevant Features. Proceedings of ECML'98: 10th European Conference on Machine Learning, Springer 1398, 137–142, (1998)

Kim, H., Howland, P. and Park, H.: Dimension Reduction in Text Classification with Support Vector Machines. Journal of Machine Learning Research, **6**, 37–53, MIT Press, (2005)

Lewis, D.D.: Reuters-21578 Text Categorization Test Collection, Distribution 1.0. http://daviddlewis.com/resources/testcollections README file (v 1.3), (2004)

Lewis, D.D., Yang, Y., Rose, T.G. and Li, F.: RCV1: A New Benchmark Collection for Text Categorization Research. Journal of Machine Learning Research, **5**, 361–397, (2004)

Porter, M.F.: An Algorithm for Suffix Stripping. Program, **14**(3), 130–137, (1980)

Salton, G. and McGill, M.: Introduction to Modern Information Retrieval. McGraw-Hill, New York, (1983)

Sebastiani, F.: Machine Learning in Automated Text Categorization. ACM Computing Surveys, **34**(1), 1–47, (2002)

Weiss, S.M., Apte, C., Damerau, F.J., Johnson, D.E., Oles, F.J., Goetz, T. and Hampp, T.: Maximizing Text-Mining Performance. IEEE Intelligent Systems, **14**(4),63–69, (1999)

Yang, Y. An Evaluation of Statistical Approaches to Text Categorization. Journal of Information Retrieval. Kluwer Academic Publishers, **1**,(1/2), 69–90, (1999)

Zelaia, A., Alegria, I., Arregi, O. and Sierra, B.: Analyzing the Effect of Dimensionality Reduction in Document Categorization for Basque. Proceedings of L&TC'05: 2nd Language & Technology Conference, 72–75, (2005)

Zelikovitz, S. and Hirsh, H.: Using LSI for Text Classification in the Presence of Background Text. Proceedings of CIKM'01: 10th ACM International Conference on Information and Knowledge Management, ACM Press, 113–118, (2001)

Zhang, T. and Oles, F.J.: Text Categorization Based on Regularized Linear Classification Methods. Information Retrieval, **4**(1): 5–31, Kluwer Academic Publishers, (2001)

Discourse Parsing: Learning FOL Rules based on Rich Verb Semantic Representations to automatically label Rhetorical Relations

Rajen Subba
Computer Science
University of Illinois
Chicago, IL, USA
rsubba@cs.uic.edu

Barbara Di Eugenio
Computer Science
University of Illinois
Chicago, IL, USA
bdieugen@cs.uic.edu

Su Nam Kim
Department of CSSE
University of Melbourne
Carlton, VIC, Australia
snkim@csse.unimelb.edu.au

Abstract

We report on our work to build a discourse parser (SemDP) that uses semantic features of sentences. We use an Inductive Logic Programming (ILP) System to exploit rich verb semantics of clauses to induce rules for discourse parsing. We demonstrate that ILP can be used to learn from highly structured natural language data and that the performance of a discourse parsing model that only uses semantic information is comparable to that of the state of the art syntactic discourse parsers.

1 Introduction

The availability of corpora annotated with syntactic information have facilitated the use of probabilistic models on tasks such as syntactic parsing. Current state of the art syntactic parsers reach accuracies between 86% and 90%, as measured by different types of precision and recall (for more details see (Collins, 2003)). Recent semantic (Kingsbury and Palmer, 2002) and discourse (Carlson et al., 2003) annotation projects are paving the way for developments in semantic and discourse parsing as well. However unlike syntactic parsing, significant development in discourse parsing remains at large.

Previous work on discourse parsing ((Soricut and Marcu, 2003) and (Forbes et al., 2001)) have focused on syntactic and lexical features only. However, discourse relations connect clauses/sentences, hence, descriptions of events and states. It makes linguistic sense that the semantics of the two clauses —generally built around the semantics of the verbs, composed with that of their arguments— affects the discourse relation(s) connecting the clauses. This may be even more evident in our instructional domain, where relations derived from planning such as *Precondition-Act* may relate clauses.

Of course, since semantic information is hard to come by, it is not surprising that previous work on discourse parsing did not use it, or only used shallow word level ontological semantics as specified in WordNet (Polanyi et al., 2004). But when rich sentence level semantics is available, it makes sense to experiment with it for discourse parsing.

A second major difficulty with using such rich verb semantic information, is that it is represented using complex data structures. Traditional Machine Learning methods cannot handle highly structured data such as First Order Logic (FOL), a representation that is suitably used to represent sentence level semantics. Such FOL representations cannot be reduced to a vector of attribute/value pairs as the relations/interdependencies that exist among the predicates would be lost.

Inductive Logic Programming (ILP) can learn structured descriptions since it learns FOL descriptions. In this paper, we present our first steps using ILP to learn semantic descriptions of discourse relations. Also of relevance to the topic of this workshop, is that discourse structure is inherently highly structured, since discourse structure is generally described in hierarchical terms: basic units of analysis, generally clauses, are related by discourse relations, resulting in more complex units, which in turn can be related via discourse relations. At the moment, we do not yet address the problem of parsing at higher levels of discourse. We intend to build on the work we present in this paper to achieve that goal.

The task of discourse parsing can be divided into two disjoint sub-problems ((Soricut and Marcu, 2003) and (Polanyi et al., 2004)). The two sub-problems are automatic identification of segment boundaries and the labeling of rhetorical relations. Though we consider the problem of automatic segmentation to be an important part in discourse parsing, we have focused entirely on the latter problem of automatically labeling rhetorical

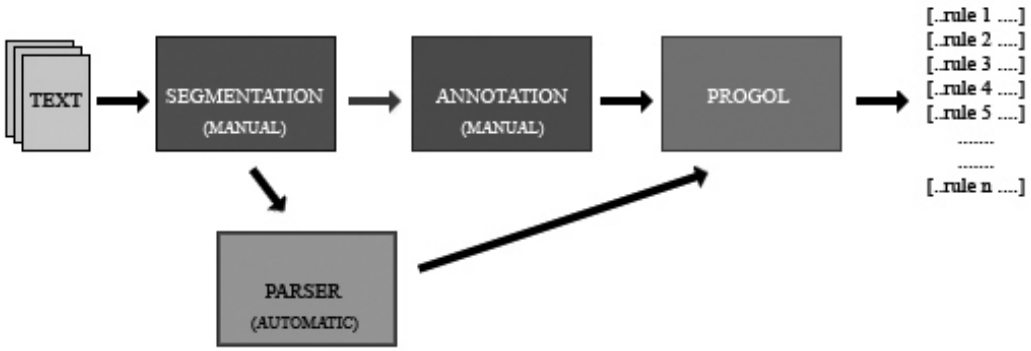


Figure 1: SemDP System Architecture (Discourse Parser)

relations only. Our approach uses rich verb semantics[1] of elementary discourse units (EDUs)[2] based on VerbNet(Kipper et al., 2000) as background knowledge and manually annotated rhetorical relations as training examples. It is trained on a lot fewer examples than the state of the art syntax-based discourse parser (Soricut and Marcu, 2003). Nevertheless, it achieves a comparable level of performance with an F-Score of 60.24. Figure 1 shows a block diagram of SemDP's system architecture. Segmentation, annotation of rhetorical relations and parsing constitute the data collection phase of the system. Learning is accomplished using an ILP based system, Progol (Muggleton, 1995). As can be seen in Figure 1, Progol takes as input both rich verb semantic information of pairs of EDUs and the rhetorical relations between them. The goal was to learn rules using the semantic information from pairs of EDUs as in Example 1:

(1) EDU1: "Sometimes, you can add a liquid to the water
EDU2: "to hasten the process"
relation(EDU1,EDU2,"Act:goal").

to automatically label unseen examples with the correct rhetorical relation.

The rest of the paper is organized as follows. Section 2 describes our data collection methodology. In section 3, Progol, the ILP system that we used to induce rules for discourse parsing is detailed. Evaluation results are presented in section 4 followed by the conclusion in section 5.

2 Data Collection

The lack of corpora annotated with both rhetorical relations as well as sentence level semantic representation led us to create our own corpus. Resources such as (Kingsbury and Palmer, 2002) and (Carlson et al., 2003) have been developed manually. Since such efforts are time consuming and costly, we decided to semi-automatically build our annotated corpus. We used an existing corpus of instructional text that is about 9MB in size and is made up entirely of written English instructions. The two largest components are home repair manuals (5Mb) and cooking recipes (1.7Mb). [3]

Segmentation. The segmentation of the corpus was done manually by a human coder. Our segmentation rules are based on those defined in (Mann and Thompson, 1988). For example, (as shown in Example 2) we segment sentences in which a conjunction is used with a clause at the conjunction site.

(2) You can copy files (//) as well as cut messages.

(//) is the segmentation marker. Sentences are segmented into EDUs. Not all the segmentation

[1]The semantic information we used is composed of VerbNet semantic predicates that capture event semantics as well as thematic roles.

[2]EDUs are minimal discourse units produced as a result of discourse segmentation.

[3]It was collected opportunistically off the internet and from other sources, and originally assembled at the Information Technology Research Institute, University of Brighton.

rules from (Mann and Thompson, 1988) are imported into our coding scheme. For example, we do not segment relative clauses. In total, our segmentation resulted in 10,084 EDUs. The segmented EDUs were then annotated with rhetorical relations by the human coder[4] and also forwarded to the parser as they had to be annotated with semantic information.

2.1 Parsing of Verb Semantics

We integrated LCFLEX (Rosé and Lavie, 2000), a robust left-corner parser, with VerbNet (Kipper et al., 2000) and CoreLex (Buitelaar, 1998). Our interest in decompositional theories of lexical semantics led us to base our semantic representation on VerbNet.

VerbNet operationalizes Levin's work and accounts for 4962 distinct verbs classified into 237 main classes. Moreover, VerbNet's strong syntactic components allow it to be easily coupled with a parser in order to automatically generate a semantically annotated corpus.

To provide semantics for nouns, we use CoreLex (Buitelaar, 1998), in turn based on the generative lexicon(Pustejovsky, 1991). CoreLex defines basic types such as *art (artifact)* or *com (communication*). Nouns that share the same bundle of basic types are grouped in the same Systematic Polysemous Class (SPC). The resulting 126 SPCs cover about 40,000 nouns.

We modified and augmented LCFLEX's existing lexicon to incorporate VerbNet and CoreLex. The lexicon is based on COMLEX (Grishman et al., 1994). Verb and noun entries in the lexicon contain a link to a semantic type defined in the ontology. VerbNet classes (including subclasses and frames) and CoreLex SPCs are realized as types in the ontology. The deep syntactic roles are mapped to the thematic roles, which are defined as variables in the ontology types. For more details on the parser see (Terenzi and Di Eugenio, 2003).

Each of the 10,084 EDUs was parsed using the parser. The parser generates both a syntactic tree and the associated semantic representation – for the purpose of this paper, we only focus on the latter. Figure 2 shows the semantic representation generated for EDU1 from Example 1, "sometimes, you can add a liquid to the water".

The semantic representation in Figure 2 is part

[4]Double annotation and segmentation is currently being done to assess inter-annotator agreement using kappa.

```
(*SEM*
 ((AGENT YOU)
  (VERBCLASS ((VNCLASS MIX-22.1-2))) (EVENT +)
  (EVENT0
   ((END
    ((ARG1 (LIQUID))
     (FRAME *TOGETHER) (ARG0 PHYSICAL)
     (ARG2 (WATER)))))))
  (EVENTSEM
   ((FRAME *CAUSE) (ARG1 E) (ARG0 (YOU)))))
  (PATIENT1 LIQUID)
  (PATIENT2 WATER)
 (ROOT-VERB ADD))
```

Figure 2: Parser Output (Semantic Information)

of the F-Structure produced by the parser. The verb *add* is parsed for a transitive frame with a PP modifier that belongs to the verb class *'MIX-22.1-2'*. The sentence contains two *PATIENT*s, namely *liquid* and *water*. *you* is identified as the *AGENT* by the parser. **TOGETHER* and **CAUSE* are the primitive semantic predicates used by VerbNet. Verb Semantics in VerbNet are defined as events that are decomposed into stages, namely start, end, during and result. The semantic representation in Figure 2 states that there is an event *EVENT0* in which the two *PATIENT*s are together at the end.

An independent evaluation on a set of 200 sentences from our instructional corpus was conducted. [5] It was able to generate complete parses for 72.2% and partial parses for 10.9% of the verb frames that we expected it to parse, given the resources. The parser cannot parse those sentences (or EDUs) that contain a verb that is not covered by VerbNet. This coverage issue, coupled with parser errors, exacerbates the problem of data sparseness. This is further worsened by the fact that we require both the EDUs in a relation set to be parsed for the Machine Learning part of our work. Addressing data sparseness is an issue left for future work.

2.2 Annotation of Rhetorical Relations

The annotation of rhetorical relations was done manually by a human coder. Our coding scheme builds on Relational Discourse Analysis (RDA) (Moser and Moore, 1995), to which we made mi-

[5]The parser evaluation was not based on EDUs but rather on unsegmented sentences. A sentence contained one or more EDUs.

nor modifications; in turn, as far as discourse relations are concerned, RDA was inspired by Rhetorical Structure Theory (RST) (Mann and Thompson, 1988).

Rhetorical relations were categorized as *informational*, *elaborational*, *temporal* and *others*. *Informational* relations describe how contents in two relata are related in the domain. These relations are further subdivided into two groups; *causality* and *similarity*. The former group consists of relations between an action and other actions or between actions and their conditions or effects. Relations like '*act:goal*', '*criterion:act*' fall under this group. The latter group consists of relations between two EDUs according to some notion of similarity such as '*restatement*' and '*contrast1:contrast2*'. *Elaborational* relations are interpropositional relations in which a proposition(s) provides detail relating to some aspect of another proposition (Mann and Thompson, 1988). Relations like '*general:specific*' and '*circumstance:situation*' belong to this category. Temporal relations like '*before:after*' capture time differences between two EDUs. Lastly, the category *others* includes relations not covered by the previous three categories such as '*joint*' and '*indeterminate*'.

Based on the modified coding scheme manual, we segmented and annotated our instructional corpus using the augmented RST tool from (Marcu et al., 1999). The RST tool was modified to incorporate our relation set. Since we were only interested in rhetorical relations that spanned between two adjacent EDUs [6], we obtained 3115 sets of potential relations from the set of all relations that we could use as training and testing data.

The parser was able to provide complete parses for both EDUs in 908 of the 3115 relation sets. These constitute the training and test set for Progol.

The semantic representation for the EDUs along with the manually annotated rhetorical relations were further processed (as shown in Figure 4) and used by Progol as input.

3 The Inductive Logic Programming Framework

We chose to use Progol, an Inductive Logic Programming system (ILP), to learn rules based on the data we collected. ILP is an area of research at the intersection of Machine Learning (ML) and Logic Programming. The general problem specification in ILP is given by the following property:

$$B \wedge H \models E \tag{3}$$

Given the background knowledge *B* and the examples *E*, ILP systems find the simplest consistent hypothesis *H*, such that *B* and *H* entails *E*.

While most of the work in NLP that involves learning has used more traditional ML paradigms like decision-tree algorithms and SVMs, we did not find them suitable for our data which is represented as Horn clauses. The requirement of using a ML system that could handle first order logic data led us to explore ILP based systems of which we found Progol most appropriate.

Progol combines Inverse Entailment with general-to-specific search through a refinement graph. A most specific clause is derived using mode declarations along with Inverse Entailment. All clauses that subsume the most specific clause form the hypothesis space. An A*-like search is used to search for the most probable theory through the hypothesis space. Progol allows arbitrary programs as background knowledge and arbitrary definite clauses as examples.

3.1 Learning from positive data only

One of the features we found appealing about Progol, besides being able to handle first order logic data, is that it can learn from positive examples alone.

Learning in natural language is a universal human process based on positive data only. However, the usual traditional learning models do not work well without negative examples. On the other hand, negative examples are not easy to obtain. Moreover, we found learning from positive data only to be a natural way to model the task of discourse parsing.

To make the learning from positive data only feasible, Progol uses a Bayesian framework. Progol learns logic programs with an arbitrarily low expected error using only positive data. Of course, we could have synthetically labeled examples of relation sets (pairs of EDUs), that did not belong to a particular relation, as negative examples. We plan to explore this approach in the future.

A key issue in learning from positive data only using a Bayesian framework is the ability to learn complex logic programs. Without any

[6] At the moment, we are concerned with learning relations between two EDUs at the base level of a Discourse Parse Tree (DPT) and not at higher levels of the hierarchy.

negative examples, the simplest rule or logic program, which in our case would be a single definite clause, would be assigned the highest score as it captures the most number of examples. In order to handle this problem, Progol's scoring function exercises a trade-off between the size of the function and the generality of the hypothesis. The score for a given hypothesis is calculated according to formula 4.

$$ln\, p(H \mid E) = m\, ln\left(\frac{1}{g(H)}\right) - sz(H) + d_m \quad (4)$$

$sz(H)$ and $g(H)$ computes the size of the hypothesis and the its generality respectively. The size of a hypothesis is measured as the number of atoms in the hypothesis whereas generality is measured by the number of positive examples the hypothesis covers. m is the number of examples covered by the hypothesis and d_m is a normalizing constant. The function $ln\ p(H|E)$ decreases with increases in $sz(H)$ and $g(H)$. As the number of examples covered (m) grow, the requirements on $g(H)$ become even stricter. This property facilitates the ability to learn more complex rules as they are supported by more positive examples. For more information on Progol and the computation of Bayes' posterior estimation, please refer to (Muggleton, 1995).

3.2 Discourse Parsing with Progol

We model the problem of assigning the correct rhetorical relation as a classification task within the ILP framework. The rich verb semantic representation of pairs of EDUs, as shown in Figure 3 [7], form the background knowledge and the manually annotated rhetorical relations between the pairs of EDUs, as shown in Figure 4, serve as the positive examples in our learning framework. The numbers in the definite clauses are *ids* used to identify the EDUs.

Progol constructs logic programs based on the background knowledge and the examples in Figures 3 and 4. Mode declarations in the Progol input file determines which clause to be used as the head (i.e. modeh) and which ones to be used in the body (i.e. modeb) of the hypotheses. Figure 5 shows an abridged set of our mode declarations.

[7] The output from the parser was further processed into definite clauses.

```
...
agent(97,you).
together(97,event0,end,physical,liquid,water).
cause(97,you,e).
patient1(97,liquid).
patient2(97,water).

theme(98,process).
rushed(98,event0,during,process).
cause(98,AGENT98,e).
...
```

Figure 3: Background Knowledge for Example 1

```
...
relation(18,19,'Act:goal').
relation(97,98,'Act:goal').
relation(1279,1280,'Step1:step2').
relation(1300,1301,'Step1:step2').
relation(1310,1311,'Step1:step2').
relation(412,413,'Before:after').
relation(441,442,'Before:after').
...
```

Figure 4: Positive Examples

Our mode declarations dictate that the predicate relation be used as the head and the other predicates (has_possession, transfer and visible) form the body of the hypotheses. '*' indicates that the number of hypotheses to learn for a given relation is unlimited. '+' and '-' signs indicate variables within the predicates of which the former is an input variable and the latter an output variable. '#' is used to denote a constant. Each argument of the predicate is a type, whether a constant or a variable. Types are defined as a single definite clause.

Our goal is to learn rules where the LHS of the rule contains the relation that we wish to learn and

```
:- modeh(*,relation(+edu,+edu,#relationtype))?

:- modeb(*,has_possession(+edu,#event,
          #eventstage,+verbarg,+verbarg))?
:- modeb(*,has_possession(+edu,#event,
          #eventstage,+verbarg,-verbarg))?
:- modeb(*,transfer(+edu,#event,#eventstage,-verbarg))?
:- modeb(*,visible(+edu,#event,#eventstage,+verbarg))?
:- modeb(*,together(+edu,#event,
          #eventstage,+verbarg,+verbarg,+verbarg))?
:- modeb(*,rushed(+edu,#event,#eventstage,+verbarg))?
```

Figure 5: Mode Declarations

```
RULE1:
 relation(EDU1,EDU2,'Act:goal') :-
   degradation_material_integrity(EDU1,event0,result,C),
   allow(EDU2,event0,during,C,D).
RULE2:
 relation(EDU1,EDU2,'Act:goal') :-
   cause(EDU1,C,D),
   together(EDU1,event0,end,E,F,G),
   cause(EDU2,C,D).
RULE3:
 relation(EDU1,EDU2,'Step1:step2') :-
   together(EDU2,event0,end,C,D,E),
   has_possession(EDU1,event0,during,C,F).
RULE4:
 relation(EDU1,EDU2,'Before:after') :-
   motion(EDU1,event0,during,C),
   location(EDU2,event0,start,C,D).
RULE6:
 relation(EDU1,EDU2,'Act:goal') :-
   motion(EDU1,event0,during,C).
```

Figure 6: Rules Learned

the RHS is a CNF of the semantic predicates defined in VerbNet with their arguments. Given the amount of training data we have, the nature of the data itself and the Bayesian framework used, Progol learns simple rules that contain just one or two clauses on the RHS. 6 of the 68 rules that Progol manages to learn are shown in Figure 6. RULE4 states that there is a theme in motion during the event in EDU A (which is the first EDU) and that the theme is located in location D at the start of the event in EDU B (the second EDU). RULE2 is learned from pairs of EDUs such as in Example 1. The simple rules in Figure 6 may not readily appeal to our intuitive notion of what such rules should include. It is not clear at this point as to how elaborate these rules should be, in order to correctly identify the relation in question. One of the reasons why more complex rules are not learned by Progol is that there aren't enough training examples. As we add more training data in the future, we will see if rules that are more elaborate than the ones in Figure 6 are learned .

4 Evaluation of the Discourse Parser

Table 1 shows the sets of relations for which we managed to obtain semantic representations (i.e. for both the EDUs).

Relations like *Preparation:act* did not yield any examples that could potentially be used. For a number of relations, the total number of examples we could use were less than 50. For the time being, we decided to use only those relation sets that had more than 50 examples. In addition, we chose not to use *Joint* and *General:specific* relations. They will be included in the future. Hence, our training and testing data consisted of the following four relations: *Goal:act*, *Step1:step2*, *Criterion:act* and *Before:after*. The total number of examples we used was 508 of which 423 were used for training and 85 were used for testing.

Relation	Total	Train Set	Test Set
Step1:step2:	**232**	**188**	**44**
Joint:	190		
Goal:act:	**170**	**147**	**23**
General:specific:	77		
Criterion:act:	**53**	**46**	**7**
Before:after:	**53**	**42**	**11**
Act:side-effect:	38		
Co-temp1:co-temp2:	22		
Cause:effect:	19		
Prescribe-act:wrong-act:	14		
Obstacle:situation:	11		
Reason:act:	9		
Restatement:	6		
Contrast1:contrast2:	6		
Circumstance:situation:	3		
Act:constraint:	2		
Criterion:wrong-act:	2		
Set:member:	1		
Act:justification:	0		
Comparison:	0		
Preparation:act:	0		
Object:attribute:	0		
Part:whole:	0		
Same-unit:	0		
Indeterminate:	0		
	908	423	85

Table 1: Relation Set Count (Total Counts include examples that yielded semantic representations for both EDUs)

Table 2, Table 3 and Table 4 show the results from running the system on our test data. A total of 85 positive examples were used for testing the system.

Table 2 evaluates our SemDP system against a baseline. Our baseline is the majority function, which performs at a 51.7 F-Score. SemDP outperforms the baseline by almost 10 percentage points

Discourse Parser	Precision	Recall	F-Score
SemDP	**61.7**	**58.8**	**60.24**
Baseline*	51.7	51.7	51.7

Table 2: Evaluation vs Baseline (* *our baseline is the majority function*)

Relation	Precision	Recall	F-Score
Goal:act	31.57	26.08	28.57
Step1:step2	**75**	**75**	**75**
Before:after	54.5	54.5	54.5
Criterion:act	71.4	71.4	71.4
Total	**61.7**	**58.8**	**60.24**

Table 3: Test Results for SemDP

with an F-Score of 60.24. To the best of our knowledge, we are also not aware of any work that uses rich semantic information for discourse parsing. (Polanyi et al., 2004) do not provide any evaluation results at all. (Soricut and Marcu, 2003) report that their SynDP parser achieved up to 63.8 F-Score on human-segmented test data. Our result of 60.24 F-Score shows that a Discourse Parser based purely on semantics can perform as well. However, since the corpus, the size of training data and the set of rhetorical relations we have used differ from (Soricut and Marcu, 2003), a direct comparison cannot be made.

Table 3 breaks down the results in detail for each of the four rhetorical relations we tested on. Since we are learning from positive data only and the rules we learn depend heavily on the amount of training data we have, we expected the system to be more accurate with the relations that have more training examples. As expected, SemDP did very well in labeling *Step1:step2* relations. Surprisingly though, it did not perform as well with *Goal:act*, even though it had the second highest number of training examples (147 in total). In fact, SemDP misclassified more positive test examples for *Goal:act* than *Before:after* or *Criterion:act*, relations which had almost one third the number of training examples. Overall SemDP achieved a precision of 61.7 and a Recall of 58.8.

In order to find out how the positive test examples were misclassified, we investigated the distribution of the relations classified by SemDP. Table 4 is the confusion matrix that highlights this issue. A majority of the actual *Goal:act* relations are incorrectly classified as *Step1:step1* and *Before:after*. Likewise, most of the misclassification of actual *Step1:step1* seems to labeled as *Goal:act* or *Before:after*. Such misclassification occurs because the simple rules learned by SemDP are not able to accurately distinguish cases where positive examples of two different relations share similar semantic predicates. Moreover, since we are learning using positive examples only, it is possible that a positive example may satisfy two or more rules for different relations. In such cases, the rule that has the highest score (as calculated by formula 4) is used to label the unseen example.

5 Conclusions and Future Work

We have shown that it is possible to learn First Order Logic rules from complex semantic data using an ILP based methodology. These rules can be used to automatically label rhetorical relations. Moreover, our results show that a Discourse Parser that uses only semantic information can perform as well as the state of the art Discourse Parsers based on syntactic and lexical information.

Future work will involve the use of syntactic information as well. We also plan to run a more thorough evaluation on the complete set of relations that we have used in our coding scheme. It is also important that the manual segmentation and annotation of rhetorical relations be subject to inter-annotator agreement. A second human annotator is currently annotating a sample of the annotated corpus. Upon completion, the annotated corpus will be checked for reliability.

Data sparseness is a well known problem in Machine Learning. Like most paradigms, our learning model is also affected by it. We also plan to explore techniques to deal with this issue.

Relation	Goal:act	Step1:step2	Before:after	Criterion:act
Goal:act	6	**8**	5	0
Step1:step2	6	**33**	5	0
Before:after	0	4	**6**	1
Criterion:act	0	0	2	**5**

Table 4: Confusion Matrix for SemDP Test Result

Lastly, we have not tackled the problem of discourse parsing at higher levels of the DPT and segmentation in this paper. Our ultimate goal is to build a Discourse Parser that will automatically segment a full text as well as annotate it with rhetorical relations at every level of the DPT using semantic as well as syntactic information. Much work needs to be done but we are excited to see what the aforesaid future work will yield.

Acknowledgments

This work is supported by award 0133123 from the National Science Foundation. Thanks to C.P. Rosé for LCFLEX, M. Palmer and K. Kipper for VerbNet, C. Buitelaar for CoreLex, and Stephen Muggleton for Progol.

References

Paul Buitelaar. 1998. *CoreLex: Systematic Polysemy and Underspecification*. Ph.D. thesis, Computer Science, Brandeis University, February.

Lynn Carlson, Daniel Marcu, and Mary Ellen Okurowski. 2003. Building a discourse-tagged corpus in the framework of Rhetorical Structure Theory. *In Current Directions in Discourse and Dialogue*, pp. 85-112, Jan van Kuppevelt and Ronnie Smith eds., Kluwer Academic Publishers.

Michael Collins. 2003. Head-driven statistical methods for natural language parsing. *Computational Linguistics*, 29.

Katherine Forbes, Eleni Miltsakaki, Rashmi Prasad, Anoop Sarkar, Aravind Joshi and Bonnie Webber. 2001. D-LTAG System - Discourse Parsing with a Lexicalized Tree Adjoining Grammar. *Information Stucture, Discourse Structure and Discourse Semantics*, ESSLLI, 2001.

Ralph Grishman, Catherine Macleod, and Adam Meyers. 1994. COMLEX syntax: Building a computational lexicon. In *COLING 94, Proceedings of the 15th International Conference on Computational Linguistics*, pages 472–477, Kyoto, Japan, August.

Paul Kingsbury and Martha Palmer. 2000. From Treebank to Propbank. In *Third International Conference on Language Resources and Evaluation, LREC-02*, Las Palmas, Canary Islands, Spain, May 28 - June 3, 2002.

Karin Kipper, Hoa Trang Dang, and Martha Palmer. 2000. Class-based construction of a verb lexicon. In *AAAI-2000, Proceedings of the Seventeenth National Conference on Artificial Intelligence*, Austin, TX.

Beth Levin and Malka Rappaport Hovav. 1992. Wiping the slate clean: a lexical semantic exploration. In Beth Levin and Steven Pinker, editors, *Lexical and Conceptual Semantics, Special Issue of Cognition: International Journal of Cognitive Science*. Blackwell Publishers.

William C. Mann and Sandra Thompson. 1988. Rhetorical Structure Theory: toward a Functional Theory of Text Organization. *Text*, 8(3):243–281.

Daniel Marcu and Abdessamad Echihabi. 2002. An unsupervised approach to recognizing discourse relations. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL-2002)*, Philadelphia, PA, July.

Daniel Marcu, Magdalena Romera and Estibaliz Amorrortu. 1999. Experiments in Constructing a Corpus of Discourse Trees: Problems, Annotation Choices, Issues. In *The Workshop on Levels of Representation in Discourse, pages 71-78*, Edinburgh, Scotland, July.

M. G. Moser, and J. D. Moore. 1995. Using Discourse Analysis and Automatic Text Generation to Study Discourse Cue Usage. In *AAAI Spring Symposium on Empirical Methods in Discourse Interpretation and Generation*, 1995.

Stephen H. Muggleton. 1995. Inverse Entailment and Progol. In *New Generation Computing Journal*, Vol. 13, pp. 245-286, 1995.

Martha Palmer, Daniel Gildea and, Paul Kingsbury. 2005. The Proposition Bank: An Annotated Corpus of Semantic Roles. *Computational Linguistics*, 31(1):71–105.

Livia Polanyi, Christopher Culy, Martin H. van den Berg, Gian Lorenzo Thione, and David Ahn. 2004. Sentential Structure and Discourse Parsing. *Proceedings of the ACL2004 Workshop on Discourse Annotation*, Barcelona, Spain, July 25, 2004.

James Pustejovsky. 1991. The generative lexicon. *Computational Linguistics*, 17(4):409–441.

Carolyn Penstein Rosé and Alon Lavie. 2000. Balancing robustness and efficiency in unification-augmented context-free parsers for large practical applications. In Jean-Clause Junqua and Gertjan van Noord, editors, *Robustness in Language and Speech Technology*. Kluwer Academic Press.

Radu Soricut and Daniel Marcu. 2003. Sentence Level Discourse Parsing using Syntactic and Lexical Information. In *Proceedings of the Human Language Technology and North American Assiciation for Computational Linguistics Conference (HLT/NAACL-2003)*, Edmonton, Canada, May-June.

Elena Terenzi and Barbara Di Eugenio. 2003. Building lexical semantic representations for natural language instructions. In *HLT-NAACL03, 2003 Human Language Technology Conference*, pages 100–102, Edmonton, Canada, May. (Short Paper).

Reranking Translation Hypotheses Using Structural Properties

Saša Hasan, Oliver Bender, Hermann Ney
Chair of Computer Science VI
RWTH Aachen University
D-52056 Aachen, Germany
{hasan,bender,ney}@cs.rwth-aachen.de

Abstract

We investigate methods that add syntactically motivated features to a statistical machine translation system in a reranking framework. The goal is to analyze whether shallow parsing techniques help in identifying ungrammatical hypotheses. We show that improvements are possible by utilizing supertagging, lightweight dependency analysis, a link grammar parser and a maximum-entropy based chunk parser. Adding features to n-best lists and discriminatively training the system on a development set increases the BLEU score up to 0.7% on the test set.

1 Introduction

Statistically driven machine translation systems are currently the dominant type of system in the MT community. Though much better than traditional rule-based approaches, these systems still make a lot of errors that seem, at least from a human point of view, illogical.

The main purpose of this paper is to investigate a means of identifying ungrammatical hypotheses from the output of a machine translation system by using grammatical knowledge that expresses syntactic dependencies of words or word groups. We introduce several methods that try to establish this kind of linkage between the words of a hypothesis and, thus, determine its well-formedness, or "fluency". We perform rescoring experiments that rerank n-best lists according to the presented framework.

As methodologies deriving well-formedness of a sentence we use supertagging (Bangalore and Joshi, 1999) with lightweight dependency analysis (LDA)[1] (Bangalore, 2000), link grammars (Sleator and Temperley, 1993) and a maximum-entropy (ME) based chunk parser (Bender et al., 2003). The former two approaches explicitly model the syntactic dependencies between words. Each hypothesis that contains irregularities, such as broken linkages or non-satisfied dependencies, should be penalized or rejected accordingly. For the ME chunker, the idea is to train n-gram models on the chunk or POS sequences and directly use the log-probability as feature score.

In general, these concepts and the underlying programs should be robust and fast in order to be able to cope with large amounts of data (as it is the case for n-best lists). The experiments presented show a small though consistent improvement in terms of automatic evaluation measures chosen for evaluation. BLEU score improvements, for instance, lie in the range from 0.3 to 0.7% on the test set.

In the following, Section 2 gives an overview on related work in this domain. In Section 3 we review our general approach to statistical machine translation (SMT) and introduce the main methodologies used for deriving syntactic dependencies on words or word groups, namely supertagging/LDA, link grammars and ME chunking. The corpora and the experiments are discussed in Section 4. The paper is concluded in Section 5.

2 Related work

In (Och et al., 2004), the effects of integrating syntactic structure into a state-of-the-art statistical machine translation system are investigated. The approach is similar to the approach presented here:

[1] In the context of this work, the term LDA is not to be confused with *linear discriminant analysis*.

firstly, a word graph is generated using the baseline SMT system and n-best lists are extracted accordingly, then additional feature functions representing syntactic knowledge are added and the corresponding scaling factors are trained discriminatively on a development n-best list.

Och and colleagues investigated a large amount of different feature functions. The field of application varies from simple syntactic features, such as IBM model 1 score, over shallow parsing techniques to more complex methods using grammars and intricate parsing procedures. The results were rather disappointing. Only one of the simplest models, i.e. the implicit syntactic feature derived from IBM model 1 score, yielded consistent and significant improvements. All other methods had only a very small effect on the overall performance.

3 Framework

In the following sections, the theoretical framework of statistical machine translation using a direct approach is reviewed. We introduce the supertagging and lightweight dependency analysis approach, link grammars and maximum-entropy based chunking technique.

3.1 Direct approach to SMT

In statistical machine translation, the best translation $\hat{e}_1^{\hat{I}} = \hat{e}_1 \dots \hat{e}_i \dots \hat{e}_{\hat{I}}$ of source words $f_1^J = f_1 \dots f_j \dots f_J$ is obtained by maximizing the conditional probability

$$\begin{aligned} \hat{e}_1^{\hat{I}} &= \underset{I,e_1^I}{\operatorname{argmax}} \{Pr(e_1^I|f_1^J)\} \\ &= \underset{I,e_1^I}{\operatorname{argmax}} \{Pr(f_1^J|e_1^I) \cdot Pr(e_1^I)\} \end{aligned} \quad (1)$$

using Bayes decision rule. The first probability on the right-hand side of the equation denotes the translation model whereas the second is the target language model.

An alternative to this classical source-channel approach is the direct modeling of the posterior probability $Pr(e_1^I|f_1^J)$ which is utilized here. Using a log-linear model (Och and Ney, 2002), we obtain

$$Pr(e_1^I|f_1^J) = \frac{\exp\left(\sum_{m=1}^M \lambda_m h_m(e_1^I, f_1^J)\right)}{\sum_{e'^{I'}_1} \exp\left(\sum_{m=1}^M \lambda_m h_m(e'^{I'}_1, f_1^J)\right)}, \quad (2)$$

where λ_m are the scaling factors of the models denoted by feature functions $h_m(\cdot)$. The denominator represents a normalization factor that depends only on the source sentence f_1^J. Therefore, we can omit it during the search process, leading to the following decision rule:

$$\hat{e}_1^{\hat{I}} = \underset{I,e_1^I}{\operatorname{argmax}} \left\{ \sum_{m=1}^M \lambda_m h_m(e_1^I, f_1^J) \right\} \quad (3)$$

This approach is a generalization of the source-channel approach. It has the advantage that additional models $h(\cdot)$ can be easily integrated into the overall system. The model scaling factors λ_1^M are trained according to the maximum entropy principle, e.g., using the GIS algorithm. Alternatively, one can train them with respect to the final translation quality measured by an error criterion (Och, 2003). For the results reported in this paper, we optimized the scaling factors with respect to a linear interpolation of word error rate (WER), position-independent word error rate (PER), BLEU and NIST score using the Downhill Simplex algorithm (Press et al., 2002).

3.2 Supertagging/LDA

Supertagging (Bangalore and Joshi, 1999) uses the Lexicalized Tree Adjoining Grammar formalism (LTAG) (XTAG Research Group, 2001). Tree Adjoining Grammars incorporate a tree-rewriting formalism using elementary trees that can be combined by two operations, namely substitution and adjunction, to derive more complex tree structures of the sentence considered. Lexicalization allows us to associate each elementary tree with a lexical item called the *anchor*. In LTAGs, every elementary tree has such a lexical anchor, also called head word. It is possible that there is more than one elementary structure associated with a lexical item, as e.g. for the case of verbs with different subcategorization frames.

The elementary structures, called initial and auxiliary trees, hold all dependent elements within the same structure, thus imposing constraints on the lexical anchors in a local context. Basically, supertagging is very similar to part-of-speech tagging. Instead of POS tags, richer descriptions, namely the elementary structures of LTAGs, are annotated to the words of a sentence. For this purpose, they are called *supertags* in order to distinguish them from ordinary POS tags. The result is an "almost parse" because of the dependencies

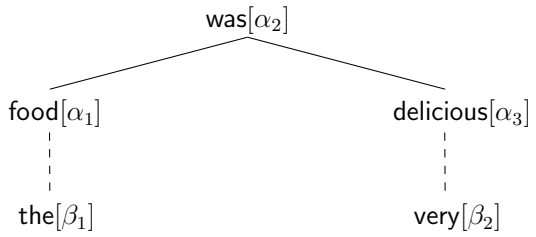


Figure 1: LDA: example of a derivation tree, β nodes are the result of the adjunction operation on auxiliary trees, α nodes of substitution on initial trees.

coded within the supertags. Usually, a lexical item can have many supertags, depending on the various contexts it appears in. Therefore, the local ambiguity is larger than for the case of POS tags. An LTAG parser for this scenario can be very slow, i.e. its computational complexity is in $O(n^6)$, because of the large number of supertags, i.e. elementary trees, that have to be examined during a parse. In order to speed up the parsing process, we can apply n-gram models on a supertag basis in order to filter out incompatible descriptions and thus improve the performance of the parser. In (Bangalore and Joshi, 1999), a trigram supertagger with smoothing and back-off is reported that achieves an accuracy of 92.2% when trained on one million running words.

There is another aspect to the dependencies coded in the elementary structures. We can use them to actually derive a shallow parse of the sentence in linear time. The procedure is presented in (Bangalore, 2000) and is called *lightweight dependency analysis*. The concept is comparable to *chunking*. The lightweight dependency analyzer (LDA) finds the arguments for the encoded dependency requirements. There exist two types of *slots* that can be filled. On the one hand, nodes marked for substitution (in α-trees) have to be filled by the complements of the lexical anchor. On the other hand, the foot nodes (i.e. nodes marked for adjunction in β-trees) take words that are being modified by the supertag. Figure 1 shows a tree derived by LDA on the sentence *the food was very delicious* from the C-Star'03 corpus (cf. Section 4.1).

The supertagging and LDA tools are available from the XTAG research group website.[2]

As features considered for the reranking experiments we choose:

[2] http://www.cis.upenn.edu/~xtag/

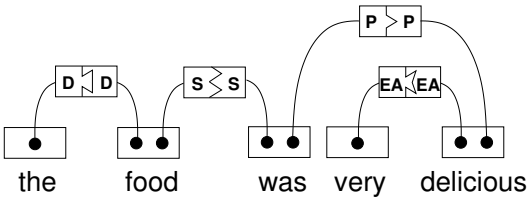


Figure 2: Link grammar: example of a valid linkage satisfying all constraints.

- Supertagger output: directly use the log-likelihoods as feature score. This did not improve performance significantly, so the model was discarded from the final system.
- LDA output:
  - dependency coverage: determine the number of covered elements, i.e. where the dependency slots are filled to the left and right
  - separate features for the number of modifiers and complements determined by the LDA

3.3 Link grammar

Similar to the ideas presented in the previous section, link grammars also explicitly code dependencies between words (Sleator and Temperley, 1993). These dependencies are called *links* which reflect the local requirements of each word. Several constraints have to be satisfied within the link grammar formalism to derive correct linkages, i.e. sets of links, of a sequence of words:

1. Planarity: links are not allowed to cross each other
2. Connectivity: links suffice to connect all words of a sentence
3. Satisfaction: linking requirements of each word are satisfied

An example of a valid linkage is shown in Figure 2. The link grammar parser that we use is freely available from the authors' website.[3] Similar to LTAG, the link grammar formalism is lexicalized which allows for enhancing the methods with probabilistic n-gram models (as is also the case for supertagging). In (Lafferty et al., 1992), the link grammar is used to derive a new class of

[3] http://www.link.cs.cmu.edu/link/

[NP the food] [VP was] [ADJP very delicious]

the/DT food/NN was/VBD very/RB delicious/JJ

Figure 3: Chunking and POS tagging: a tag next to the opening bracket denotes the type of chunk, whereas the corresponding POS tag is given after the word.

language models that, in comparison to traditional n-gram LMs, incorporate capabilities for expressing long-range dependencies between words.

The link grammar dictionary that specifies the words and their corresponding valid links currently holds approximately 60 000 entries and handles a wide variety of phenomena in English. It is derived from newspaper texts.

Within our reranking framework, we use link grammar features that express a possible well-formedness of the translation hypothesis. The simplest feature is a binary one stating whether the link grammar parser could derive a complete linkage or not, which should be a strong indicator of a syntactically correct sentence. Additionally, we added a normalized cost of the matching process which turned out not to be very helpful for rescoring, so it was discarded.

3.4 ME chunking

Like the methods described in the two preceding sections, text chunking consists of dividing a text into syntactically correlated non-overlapping groups of words. Figure 3 shows again our example sentence illustrating this task. Chunks are represented as groups of words between square brackets. We employ the 11 chunk types as defined for the CoNLL-2000 shared task (Tjong Kim Sang and Buchholz, 2000).

For the experiments, we apply a maximum-entropy based tagger which has been successfully evaluated on natural language understanding and named entity recognition (Bender et al., 2003). Within this tool, we directly factorize the posterior probability and determine the corresponding chunk tag for each word of an input sequence. We assume that the decisions depend only on a limited window $e_{i-2}^{i+2} = e_{i-2}...e_{i+2}$ around the current word e_i and on the two predecessor chunk tags c_{i-2}^{i-1}. In addition, part-of-speech (POS) tags g_1^I are assigned and incorporated into the model (cf. Figure 3). Thus, we obtain the following second-order model:

$$\begin{aligned} Pr(c_1^I|e_1^I, g_1^I) &= \\ &= \prod_{i=1}^{I} Pr(c_i|c_1^{i-1}, e_1^I, g_1^I) \quad (4) \\ &= \prod_{i=1}^{I} p(c_i|c_{i-2}^{i-1}, e_{i-2}^{i+2}, g_{i-2}^{i+2}), \quad (5) \end{aligned}$$

where the step from Eq. 4 to 5 reflects our model assumptions.

Furthermore, we have implemented a set of binary valued feature functions for our system, including lexical, word and transition features, prior features, and compound features, cf. (Bender et al., 2003). We run simple count-based feature reduction and train the model parameters using the Generalized Iterative Scaling (GIS) algorithm (Darroch and Ratcliff, 1972). In practice, the training procedure tends to result in an overfitted model. To avoid this, a smoothing method is applied where a Gaussian prior on the parameters is assumed (Chen and Rosenfeld, 1999).

Within our reranking framework, we firstly use the ME based tagger to produce the POS and chunk sequences for the different n-best list hypotheses. Given several n-gram models trained on the WSJ corpus for both POS and chunk models, we then rescore the n-best hypotheses and simply use the log-probabilities as additional features. In order to adapt our system to the characteristics of the data used, we build POS and chunk n-gram models on the training corpus part. These domain-specific models are also added to the n-best lists.

The ME chunking approach does not model explicit syntactic linkages of words. Instead, it incorporates a statistical framework to exploit valid and syntactically coherent groups of words by additionally looking at the word classes.

4 Experiments

For the experiments, we use the translation system described in (Zens et al., 2005). Our phrase-based decoder uses several models during search that are interpolated in a log-linear way (as expressed in Eq. 3), such as phrase-based translation models, word-based lexicon models, a language, deletion and simple reordering model and word and phrase penalties. A word graph containing the most likely translation hypotheses is generated during the search process. Out of this compact

		Supplied Data Track			
		Arabic	Chinese	Japanese	English
Train	Sentences	20 000			
	Running Words	180 075	176 199	198 453	189 927
	Vocabulary	15 371	8 687	9 277	6 870
	Singletons	8 319	4 006	4 431	2 888
C-Star'03	Sentences	506			
	Running Words	3 552	3 630	4 130	3 823
	OOVs (Running Words)	133	114	61	65
IWSLT'04	Sentences	500			
	Running Words	3 597	3 681	4 131	3 837
	OOVs (Running Words)	142	83	71	58

Table 1: Corpus statistics after preprocessing.

representation, we extract n-best lists as described in (Zens and Ney, 2005). These n-best lists serve as a starting point for our experiments. The methods presented in Section 3 produce scores that are used as additional features for the n-best lists.

4.1 Corpora

The experiments are carried out on a subset of the *Basic Travel Expression Corpus* (BTEC) (Takezawa et al., 2002), as it is used for the supplied data track condition of the IWSLT evaluation campaign. BTEC is a multilingual speech corpus which contains tourism-related sentences similar to those that are found in phrase books. For the supplied data track, the training corpus contains 20 000 sentences. Two test sets, C-Star'03 and IWSLT'04, are available for the language pairs Arabic-English, Chinese-English and Japanese-English.

The corpus statistics are shown in Table 1. The average source sentence length is between seven and eight words for all languages. So the task is rather limited and very domain-specific. The advantage is that many different reranking experiments with varying feature function settings can be carried out easily and quickly in order to analyze the effects of the different models.

In the following, we use the C-Star'03 set for development and tuning of the system's parameters. After that, the IWSLT'04 set is used as a blind test set in order to measure the performance of the models.

4.2 Rescoring experiments

The use of n-best lists in machine translation has several advantages. It alleviates the effects of the huge search space which is represented in word graphs by using a compact excerpt of the n best hypotheses generated by the system. Especially for limited domain tasks, the size of the n-best list can be rather small but still yield good oracle error rates. Empirically, n-best lists should have an appropriate size such that the oracle error rate, i.e. the error rate of the best hypothesis with respect to an error measure (such as WER or PER) is approximately half the baseline error rate of the system. N-best lists are suitable for easily applying several rescoring techniques since the hypotheses are already fully generated. In comparison, word graph rescoring techniques need specialized tools which can traverse the graph accordingly. Since a node within a word graph allows for many histories, one can only apply local rescoring techniques, whereas for n-best lists, techniques can be used that consider properties of the whole sentence.

For the Chinese-English and Arabic-English task, we set the n-best list size to $n = 1500$. For Japanese-English, $n = 1000$ produces oracle error rates that are deemed to be sufficiently low, namely 17.7% and 14.8% for WER and PER, respectively. The single-best output for Japanese-English has a word error rate of 33.3% and position-independent word error rate of 25.9%.

For the experiments, we add additional features to the initial models of our decoder that have shown to be particularly useful in the past, such as IBM model 1 score, a clustered language model score and a word penalty that prevents the hypotheses to become too short. A detailed definition of these additional features is given in (Zens et al., 2005). Thus, the baseline we start with is

Chinese → English, C-Star'03	NIST	BLEU[%]	mWER[%]	mPER[%]
Baseline	8.17	46.2	48.6	41.4
with supertagging/LDA	8.29	46.5	48.4	41.0
with link grammar	8.43	45.6	47.9	41.1
with supertagging/LDA + link grammar	8.22	47.5	47.7	40.8
with ME chunker	8.65	47.3	47.4	40.4
with all models	8.42	47.0	47.4	40.5
Chinese → English, IWSLT'04	NIST	BLEU[%]	mWER[%]	mPER[%]
Baseline	8.67	45.5	49.1	39.8
with supertagging/LDA	8.68	45.4	49.8	40.3
with link grammar	8.81	45.0	49.0	40.2
with supertagging/LDA+link grammar	8.56	46.0	49.1	40.6
with ME chunker	9.00	44.6	49.3	40.6
with all models	8.89	46.2	48.1	39.6

Table 2: Effect of successively adding syntactic features to the Chinese-English n-best list for C-Star'03 (development set) and IWSLT'04 (test set).

BASE	*Any messages for me?*
RESC	*Do you have any messages for me?*
REFE	*Do you have any messages for me?*
BASE	*She, not yet?*
RESC	*She has not come yet?*
REFE	*Lenny, she has not come in?*
BASE	*How much is it to the?*
RESC	*How much is it to the local call?*
REFE	*How much is it to the city centre?*
BASE	*This blot or.*
RESC	*This is not clean.*
REFE	*This still is not clean.*

Table 3: Translation examples for the Chinese-English test set (IWSLT'04): baseline system (BASE) vs. rescored hypotheses (RESC) and reference translation (REFE).

already a very strong one. The log-linear interpolation weights λ_m from Eq. 3 are directly optimized using the Downhill Simplex algorithm on a linear combination of WER (word error rate), PER (position-independent word error rate), NIST and BLEU score.

In Table 2, we show the effect of adding the presented features successively to the baseline. Separate entries for experiments using supertagging/LDA and link grammars show that a combination of these syntactic approaches always yields some gain in translation quality (regarding BLEU score). The performance of the maximum-entropy based chunking is comparable. A combination of all three models still yields a small improvement.

Table 3 shows some examples for the Chinese-English test set. The rescored translations are syntactically coherent, though semantical correctness cannot be guaranteed. On the test data, we achieve an overall improvement of 0.7%, 0.5% and 0.3% in BLEU score for Chinese-English, Japanese-English and Arabic-English, respectively (cf. Tables 4 and 5).

4.3 Discussion

From the tables, it can be seen that the use of syntactically motivated feature functions within a reranking concept helps to slightly reduce the number of translation errors of the overall translation system. Although the improvement on the IWSLT'04 set is only moderate, the results are nevertheless comparable or better to the ones from (Och et al., 2004), where, starting from IBM model 1 baseline, an additional improvement of only 0.4% BLEU was achieved using more complex methods.

For the maximum-entropy based chunking approach, n-grams with $n = 4$ work best for the chunker that is trained on WSJ data. The domain-specific rescoring model which results from the chunker being trained on the BTEC corpora turns out to prefer higher order n-grams, with $n = 6$ or more. This might be an indicator of the domain-specific rescoring model successfully capturing more local context.

The training of the other models, i.e. supertagging/LDA and link grammar, is also performed on

Japanese → English, C-Star'03	NIST	BLEU[%]	mWER[%]	mPER[%]
Baseline	9.09	57.8	31.3	25.0
with supertagging/LDA	9.13	57.8	31.3	24.8
with link grammar	9.46	57.6	31.9	25.3
with supertagging/LDA + link grammar	9.24	58.2	31.0	24.8
with ME chunker	9.31	58.7	30.9	24.4
with all models	9.21	58.9	30.5	24.3
Japanese → English, IWSLT'04	NIST	BLEU[%]	mWER[%]	mPER[%]
Baseline	9.22	54.7	34.1	25.5
with supertagging/LDA	9.27	54.8	34.2	25.6
with link grammar	9.37	54.9	34.3	25.9
with supertagging/LDA + link grammar	9.30	55.0	34.0	25.6
with ME chunker	9.27	55.0	34.2	25.5
with all models	9.27	55.2	33.9	25.5

Table 4: Effect of successively adding syntactic features to the Japanese-English n-best list for C-Star'03 (development set) and IWSLT'04 (test set).

Arabic → English, C-Star'03	NIST	BLEU[%]	mWER[%]	mPER[%]
Baseline	10.18	64.3	23.9	20.6
with supertagging/LDA	10.13	64.6	23.4	20.1
with link grammar	10.06	64.7	23.4	20.3
with supertagging/LDA + link grammar	10.20	65.0	23.2	20.2
with ME chunker	10.11	65.1	23.0	19.9
with all models	10.23	65.2	23.0	19.9
Arabic → English, IWSLT'04	NIST	BLEU[%]	mWER[%]	mPER[%]
Baseline	9.75	59.8	26.1	21.9
with supertagging/LDA	9.77	60.5	25.6	21.5
with link grammar	9.74	60.5	25.9	21.7
with supertagging/LDA + link grammar	9.86	60.8	26.0	21.6
with ME chunker	9.71	59.9	25.9	21.8
with all models	9.84	60.1	26.4	21.9

Table 5: Effect of successively adding syntactic features to the Arabic-English n-best list for C-Star'03 (development set) and IWSLT'04 (test set).

out-of-domain data. Thus, further improvements should be possible if the models were adapted to the BTEC domain. This would require the preparation of an annotated corpus for the supertagger and a specialized link grammar, which are both time-consuming tasks.

The syntactically motivated methods (supertagging/LDA and link grammars) perform similarly to the maximum-entropy based chunker. It seems that both approaches successfully exploit structural properties of language. However, one outlier is ME chunking on the Chinese-English test data, where we observe a lower BLEU but a larger NIST score. For Arabic-English, the combination of all methods does not seem to generalize well on the test set. In that case, supertagging/LDA and link grammar outperforms the ME chunker: the overall improvement is 1% absolute in terms of BLEU score.

5 Conclusion

We added syntactically motivated features to a statistical machine translation system in a reranking framework. The goal was to analyze whether shallow parsing techniques help in identifying ungrammatical hypotheses. We showed that some improvements are possible by utilizing supertagging, lightweight dependency analysis, a link

grammar parser and a maximum-entropy based chunk parser. Adding features to n-best lists and discriminatively training the system on a development set helped to gain up to 0.7% in BLEU score on the test set.

Future work could include developing an adapted LTAG for the BTEC domain or incorporating n-gram models into the link grammar concept in order to derive a long-range language model (Lafferty et al., 1992). However, we feel that the current improvements are not significant enough to justify these efforts. Additionally, we will apply these reranking methods to larger corpora in order to study the effects on longer sentences from more complex domains.

Acknowledgments

This work has been partly funded by the European Union under the integrated project TC-Star (Technology and Corpora for Speech to Speech Translation, IST-2002-FP6-506738, http://www.tc-star.org), and by the R&D project TRAMES managed by Bertin Technologies as prime contractor and operated by the french DGA (Délégation Générale pour l'Armement).

References

Srinivas Bangalore and Aravind K. Joshi. 1999. Supertagging: An approach to almost parsing. *Computational Linguistics*, 25(2):237–265.

Srinivas Bangalore. 2000. A lightweight dependency analyzer for partial parsing. *Computational Linguistics*, 6(2):113–138.

Oliver Bender, Klaus Macherey, Franz Josef Och, and Hermann Ney. 2003. Comparison of alignment templates and maximum entropy models for natural language understanding. In *EACL03: 10th Conf. of the Europ. Chapter of the Association for Computational Linguistics*, pages 11–18, Budapest, Hungary, April.

Stanley F. Chen and Ronald Rosenfeld. 1999. A gaussian prior for smoothing maximum entropy models. Technical Report CMUCS-99-108, Carnegie Mellon University, Pittsburgh, PA.

J. N. Darroch and D. Ratcliff. 1972. Generalized iterative scaling for log-linear models. *Annals of Mathematical Statistics*, 43:1470–1480.

John Lafferty, Daniel Sleator, and Davy Temperley. 1992. Grammatical trigrams: A probabilistic model of link grammar. In *Proc. of the AAAI Fall Symposium on Probabilistic Approaches to Natural Language*, pages 89–97, Cambridge, MA.

Franz Josef Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proc. of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 295–302, Philadelphia, PA, July.

Franz Josef Och, Daniel Gildea, Sanjeev Khudanpur, Anoop Sarkar, Kenji Yamada, Alex Fraser, Shankar Kumar, Libin Shen, David Smith, Katherine Eng, Viren Jain, Zhen Jin, and Dragomir Radev. 2004. A smorgasbord of features for statistical machine translation. In *Proc. 2004 Meeting of the North American chapter of the Association for Computational Linguistics (HLT-NAACL)*, pages 161–168, Boston, MA.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proc. of the 41st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 160–167, Sapporo, Japan, July.

William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. 2002. *Numerical Recipes in C++*. Cambridge University Press, Cambridge, UK.

Daniel Sleator and Davy Temperley. 1993. Parsing English with a link grammar. In *Third International Workshop on Parsing Technologies*, Tilburg/Durbuy, The Netherlands/Belgium, August.

Toshiyuki Takezawa, Eiichiro Sumita, F. Sugaya, H. Yamamoto, and S. Yamamoto. 2002. Toward a broad-coverage bilingual corpus for speech translation of travel conversations in the real world. In *Proc. of the Third Int. Conf. on Language Resources and Evaluation (LREC)*, pages 147–152, Las Palmas, Spain, May.

Erik F. Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the CoNLL-2000 shared task: Chunking. In *Proceedings of CoNLL-2000 and LLL-2000*, pages 127–132, Lisbon, Portugal, September.

XTAG Research Group. 2001. A Lexicalized Tree Adjoining Grammar for English. Technical Report IRCS-01-03, IRCS, University of Pennsylvania, Philadelphia, PA, USA.

Richard Zens and Hermann Ney. 2005. Word graphs for statistical machine translation. In *43rd Annual Meeting of the Assoc. for Computational Linguistics: Proc. Workshop on Building and Using Parallel Texts: Data-Driven Machine Translation and Beyond*, pages 191–198, Ann Arbor, MI, June.

Richard Zens, Oliver Bender, Saša Hasan, Shahram Khadivi, Evgeny Matusov, Jia Xu, Yuqi Zhang, and Hermann Ney. 2005. The RWTH phrase-based statistical machine translation system. In *Proceedings of the International Workshop on Spoken Language Translation (IWSLT)*, pages 155–162, Pittsburgh, PA, October.

Tree Kernel Engineering in Semantic Role Labeling Systems

Alessandro Moschitti and **Daniele Pighin** and **Roberto Basili**
University of Rome, Tor Vergata
{moschitti,basili}@info.uniroma2.it
daniele.pighin@gmail.com

Abstract

Recent work on the design of automatic systems for semantic role labeling has shown that feature engineering is a complex task from a modeling and implementation point of view. Tree kernels alleviate such complexity as kernel functions generate features automatically and require less software development for data extraction.

In this paper, we study several tree kernel approaches for both boundary detection and argument classification. The comparative experiments on Support Vector Machines with such kernels on the CoNLL 2005 dataset show that very simple tree manipulations trigger automatic feature engineering that highly improves accuracy and efficiency in both phases. Moreover, the use of different classifiers for internal and pre-terminal nodes maintains the same accuracy and highly improves efficiency.

1 Introduction

A lot of attention has been recently devoted to the design of systems for the automatic labeling of semantic roles (SRL) as defined in two important projects: FrameNet (Johnson and Fillmore, 2000), inspired by Frame Semantics, and PropBank (Kingsbury and Palmer, 2002) based on Levin's verb classes. In general, given a sentence in natural language, the annotation of a predicate's semantic roles requires (1) the detection of the target word that embodies the predicate and (2) the detection and classification of the word sequences constituting the predicate's arguments. In particular, step (2) can be divided into two different phases: (a) boundary detection, in which the words of the sequence are detected and (b) argument classification, in which the type of the argument is selected.

Most machine learning models adopted for the SRL task have shown that (shallow or deep) syntactic information is necessary to achieve a good labeling accuracy. This research brings a wide empirical evidence in favor of the linking theories between semantics and syntax, e.g. (Jackendoff, 1990). However, as no theory provides a sound and complete treatment of such issue, the choice and design of syntactic features for the automatic learning of semantic structures requires remarkable research efforts and intuition.

For example, the earlier studies concerning linguistic features suitable for semantic role labeling were carried out in (Gildea and Jurasfky, 2002). Since then, researchers have proposed diverse syntactic feature sets that only slightly enhance the previous ones, e.g. (Xue and Palmer, 2004) or (Carreras and Màrquez, 2005). A careful analysis of such features reveals that most of them are syntactic tree fragments of training sentences, thus a natural way to represent them is the adoption of tree kernels as described in (Moschitti, 2004). The idea is to associate with each argument the minimal subtree that includes the target predicate with one of its arguments, and to use a tree kernel function to evaluate the number of common substructures between two such trees. Such approach is in line with current research on the use of tree kernels for natural language learning, e.g. syntactic parsing re-ranking (Collins and Duffy, 2002), relation extraction (Zelenko et al., 2003) and named entity recognition (Cumby and Roth, 2003; Culotta and Sorensen, 2004).

Regarding the use of tree kernels for SRL, in (Moschitti, 2004) two main drawbacks have been

pointed out:

- Highly accurate boundary detection cannot be carried out by a tree kernel model since correct and incorrect arguments may share a large portion of the encoding trees, i.e. they may share many substructures.
- Manually derived features (extended with a polynomial kernel) have been shown to be superior to tree kernel approaches.

Nevertheless, we believe that modeling a completely kernelized SRL system is useful for the following reasons:

- We can implement it very quickly as the feature extractor module only requires the writing of the subtree extraction procedure. Traditional SRL systems are, in contrast, based on the extraction of more than thirty features (Pradhan et al., 2005), which require the writing of at least thirty different procedures.
- Combining it with a traditional attribute-value SRL system allows us to obtain a more accurate system. Usually the combination of two traditional systems (based on the same machine learning model) does not result in an improvement as their features are more or less equivalent as shown in (Carreras and Màrquez, 2005).
- The study of the effective structural features can inspire the design of novel linear features which can be used with a more efficient model (i.e. linear SVMs).

In this paper, we carry out tree kernel engineering (Moschitti et al., 2005) to increase both accuracy and speed of the boundary detection and argument classification phases. The engineering approach relates to marking the nodes of the encoding subtrees in order to generate substructures more strictly correlated with a particular argument, boundary or predicate. For example, marking the node that exactly covers the target argument helps tree kernels to generate different substructures for correct and incorrect argument boundaries.

The other technique that we applied to engineer different kernels is the subdivision of internal and pre-terminal nodes. We show that designing different classifiers for these two different node types slightly increases the accuracy and remarkably decreases the learning and classification time.

An extensive experimentation of our tree kernels with Support Vector Machines on the CoNLL 2005 data set provides interesting insights on the design of performant SRL systems entirely based on tree kernels.

In the remainder of this paper, Section 2 introduces basic notions on SRL systems and tree kernels. Section 3 illustrates our new kernels for both boundary and classification tasks. Section 4 shows the experiments of SVMs with the above tree kernel based classifiers.

2 Preliminary Concepts

In this section we briefly define the SRL model that we intend to design and the kernel function that we use to evaluate the similarity between subtrees.

2.1 Basic SRL approach

The SRL approach that we adopt is based on the deep syntactic parse (Charniak, 2000) of the sentence that we intend to annotate semantically. The standard algorithm is to classify the tree node pair $\langle p, a\rangle$, where p and a are the nodes that exactly cover the target predicate and a potential argument, respectively. If $\langle p, a\rangle$ is labeled with an argument, then the terminal nodes dominated by a will be considered as the words constituting such argument. The number of pairs for each sentence can be hundreds, thus, if we consider training corpora of thousands of sentences, we have to deal with millions of training instances.

The usual solution to limit such complexity is to divide the labeling task in two subtasks:

- Boundary detection, in which a single classifier is trained on many instances to detect if a node is an argument or not, i.e. if the sequence of words dominated by the target node constitutes a correct boundary.
- Argument classification: only the set of nodes corresponding to correct boundaries are considered. These can be used to train a multiclassifier that, for such nodes, only decides the type of the argument. For example, we can train n classifiers in the style One-vs-All. At classification time, for each argument node, we can select the argument type associated with the maximum among the n scores provided by the single classifiers.

We adopt this solution as it enables us to use only one computationally expensive classifier, i.e. the boundary detection one. This, as well as the argument classifiers, requires a feature representation of the predicate-argument pair. Such features are mainly extracted from the parse trees of the target sentence, e.g. *Phrase Type*, *Predicate Word*, *Head Word*, *Governing Category*, *Position* and *Voice* proposed in (Gildea and Jurasfky, 2002).

As most of the features proposed in literature are subsumed by tree fragments, tree-kernel functions are a natural way to produce them automatically.

2.2 Tree kernel functions

Tree-kernel functions simply evaluate the number of substructures shared between two trees T_1 and T_2. Such functions can be seen as a scalar product in the huge vector space constituted by all possible substructures of the training set. Thus, kernel functions implicitly define a large feature space.

Formally, given a tree fragment space $\{f_1, f_2, ..\} = \mathcal{F}$, we can define an indicator function $I_i(n)$, which is equal to 1 if the target f_i is rooted at node n and equal to 0 otherwise. Therefore, a tree-kernel function K over T_1 and T_2 can be defined as $K(T_1, T_2) = \sum_{n_1 \in N_{T_1}} \sum_{n_2 \in N_{T_2}} \Delta(n_1, n_2)$, where N_{T_1} and N_{T_2} are the sets of the T_1's and T_2's nodes, respectively and $\Delta(n_1, n_2) = \sum_{i=1}^{|\mathcal{F}|} I_i(n_1) I_i(n_2)$. This latter is equal to the number of common fragments rooted at nodes n_1 and n_2 and, according to (Collins and Duffy, 2002), it can be computed as follows:

1. if the productions at n_1 and n_2 are different then $\Delta(n_1, n_2) = 0$;
2. if the productions at n_1 and n_2 are the same, and n_1 and n_2 have only leaf children (i.e. they are pre-terminal symbols) then $\Delta(n_1, n_2) = \lambda$;
3. if the productions at n_1 and n_2 are the same, and n_1 and n_2 are not pre-terminal then $\Delta(n_1, n_2) = \lambda \prod_{j=1}^{nc(n_1)} (1 + \Delta(c_{n_1}^j, c_{n_2}^j))$.

where λ is the decay factor to scale down the impact of large structures, $nc(n_1)$ is the number of the children of n_1 and c_n^j is the j-th child of the node n. Note that, as the productions are the same, $nc(n_1) = nc(n_2)$. Additionally, to map similarity scores in the [0,1] range, we applied a normalization in the kernel space, i.e. $K'(T_1, T_2) = \frac{K(T_1,T_2)}{\sqrt{K(T_1,T_1) \times K(T_2,T_2)}}$.

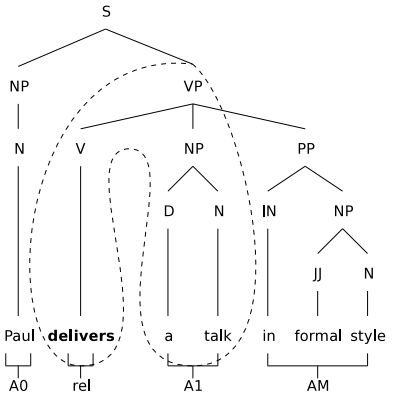


Figure 1: The PAF subtree associated with A1.

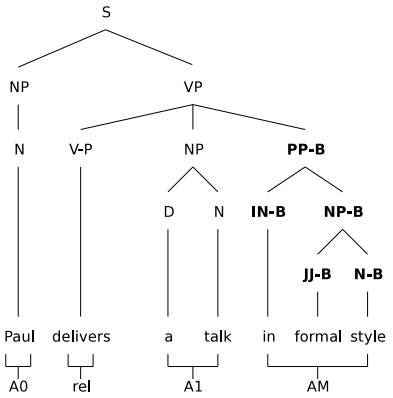


Figure 2: Example of CMST.

Once a kernel function is defined, we need to characterize the predicate-argument pair with a subtree. This allows kernel machines to generate a large number of syntactic features related to such pair. The approach proposed in (Moschitti, 2004) selects the minimal subtree that includes a predicate with its argument. We follow such approach by studying and proposing novel, interesting solutions.

3 Novel Kernels for SRL

The basic structure used to characterize the predicate argument relation is the smallest subtree that includes a predicate with one of its argument. For example, in Figure 1, the dashed line encloses a predicate argument feature (PAF) over the parse tree of the sentence: "Paul delivers a talk in formal style". This PAF is a subtree that characterizes the predicate *to deliver* with its argument *a talk*. In this section, we improve PAFs, propose different kernels for internal and pre-terminal nodes and new kernels based on complete predicate ar-

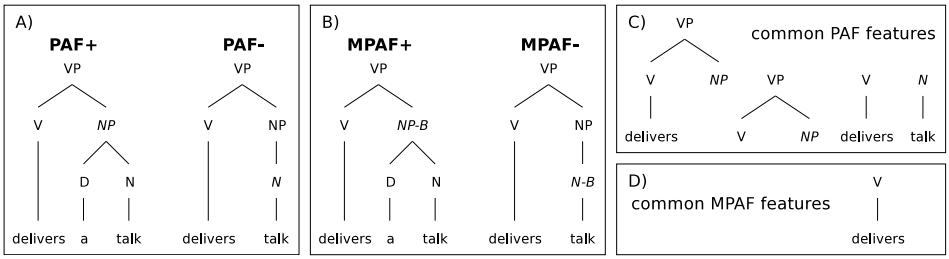


Figure 3: Differences between PAF (a) and MPAF (b) structures.

gument structures.

3.1 Improving PAF

PAFs have shown to be very effective for argument classification but not for boundary detection. The reason is that two nodes that encode correct and incorrect boundaries may generate very similar PAFs. For example, Figure 3.A shows two PAFs corresponding to a correct (PAF+) and an incorrect (PAF-) choice of the boundary for A1: PAF+ from the `NP` vs. PAF- from the `N` nodes. The number of their common substructures is high, i.e. the four subtrees shown in Frame C. This prevents the algorithm from making different decisions for such cases.

To solve this problem, we specify which is the node that exactly covers the argument (also called argument node) by simply marking it with the label `B` denoting the boundary property. Figure 3.B shows the two new marked PAFs (MPAFs). The features generated from the two subtrees are now very different so that there is only one substructure in common (see Frame D). Note that, each markup strategy impacts on the output of a kernel function in terms of the number of structures common to two trees. The same output can be obtained using unmarked trees and redefining consistently the kernel function, e.g. the algorithm described in Section 2.2.

An alternative way to partially solve the structure overlapping problem is the use of two different classifiers, one for the internal nodes and one for the pre-terminal nodes, and combining their decisions. In this way, the negative example of Figure 3 would not be used to train the same classifier that uses PAF+. Of course, similar structures can both be rooted on internal nodes, therefore they can belong to the training data of the same classifier. However, the use of different classifiers is motivated also by the fact that many argument types can be found mostly in pre-terminal nodes, e.g. modifier or negation arguments, and do not necessitate training data extracted from internal nodes. Consequently, it is more convenient (at least from a computational point of view) to use two different boundary classifiers, hereinafter referred to as combined classifier.

3.2 Kernels on complete predicate argument structures

The type of a target argument strongly depends on the type and number of the predicate's arguments[1] (Punyakanok et al., 2005; Toutanova et al., 2005). Consequently, to correctly label an argument, we should extract features from the complete predicate argument structure it belongs to. In contrast, PAFs completely neglect the information (i.e. the tree portions) related to non-target arguments.

One way to use this further information with tree kernels is to use the minimum subtree that spans all the predicate's arguments. The whole parse tree in Figure 1 is an example of such Minimum Spanning Tree (MST) as it includes all and only the argument structures of the predicate "to deliver". However, MSTs pose some problems:

- We cannot use them for the boundary detection task since we do not know the predicate's argument structure yet. However, we can derive the MST (its approximation) from the nodes selected by a boundary classifier, i.e. the nodes that correspond to potential arguments. Such approximated MSTs can be easily used in the argument type classification phase. They can also be used to re-rank the most probable m sequences of arguments for both labeling phases.

- Obviously, an MST is the same for all the arguments it includes, thus we need a way to differentiate it for each target argument.

[1] This is true at least for core arguments.

Again, we can mark the node that exactly covers the target argument as shown in the previous section. We refer to this subtree as marked MST (MMST). However, for large arguments (i.e. spread on a large part of the sentence tree) the substructures' likelihood of being part of other arguments is quite high.

To address this latter problem, we can mark all nodes that descend from the target argument node. Figure 2 shows a MST in which the subtree associated with the target argument (AM) has the nodes marked. We refer to this structure as a completely marked MST (CMST). CMSTs may be seen as PAFs enriched with new information coming from the other arguments (i.e. the non-marked subtrees). Note that if we consider only the PAF subtree from a CMST we obtain a differently marked subtree which we refer to as CPAF.

In the next section we study the impact of the proposed kernels on the boundary detection and argument classification performance.

4 Experiments

In these experiments we evaluate the impact of our proposed kernels in terms of accuracy and efficiency. The accuracy improvement confirms that the node marking approach enables the automatic engineering of effective SRL features. The efficiency improvement depends on (a) the less training data used when applying two distinct type classifiers for internal and pre-terminal nodes and (b) a more adequate feature space which allows SVMs to converge faster to a model containing a smaller number of support vectors, i.e. faster training and classification.

4.1 Experimental set up

The empirical evaluations were carried out within the setting defined in the CoNLL-2005 Shared Task (Carreras and Màrquez, 2005). We used as a target dataset the PropBank corpus available at `www.cis.upenn.edu/~ace`, along with the Penn TreeBank 2 for the gold trees (`www.cis.upenn.edu/~treebank`) (Marcus et al., 1993), which includes about 53,700 sentences. Since the aim of this study was to design a real SRL system we adopted the Charniak parse trees from the CoNLL 2005 Shared Task data (available at `www.lsi.upc.edu/~srlconll/`).

We used Section 02, 03 and 24 from the Penn TreeBank in most of the experiments. Their characteristics are shown in Table 1. *Pos* and *Neg* indicate the number of nodes corresponding or not to a correct argument boundary. Rows 3 and 4 report such number for the internal and pre-terminal nodes separately. We note that the latter are much fewer than the former; this results in a very fast pre-terminal classifier.

As the automatic parse trees contain errors, some arguments cannot be associated with any covering node. This prevents us to extract a tree representation for them. Consequently, we do not consider them in our evaluation. In sections 2, 3 and 24 there are 454, 347 and 731 such cases, respectively.

The experiments were carried out with the SVM-light-TK software available at `http://ai-nlp.info.uniroma2.it/moschitti/` which encodes fast tree kernel evaluation (Moschitti, 2006) in the SVM-light software (Joachims, 1999). We used a regularization parameter (option `-c`) equal to 1 and $\lambda = 0.4$ (see (Moschitti, 2004)).

4.2 Boundary Detection Results

In these experiments, we used Section 02 for training and Section 24 for testing. The results using the PAF and the MPAF based kernels are reported in Table 2 in rows 2 and 3, respectively. Columns 3 and 4 show the CPU testing time (in seconds) and the F_1 of the monolithic boundary classifier. The next 3 columns show the CPU time for the internal (Int) and pre-terminal (Pre) node classifiers, as well as their total (All). The F_1 measures are reported in the 3 rightmost columns. In particular, the third column refers to the F_1 of the combined classifier. This has been computed by summing correct, incorrect and not retrieved examples of the two distinct classifiers.

We note that: first, the monolithic classifier applied to MPAF improves both the efficiency, i.e. about 3,131 seconds vs. 5,179, of PAF and the F_1, i.e. 82.07 vs. 75.24. This suggests that marking the argument node simplifies the generalization process.

Second, by dividing the boundary classification in two tasks, internal and pre-terminal nodes, we furthermore improve the classification time for both PAF and MPAF kernels, i.e. 5,179 vs. 1,851 (PAF) and 3,131 vs. 1,471 (MPAF). The separated classifiers are much faster, especially the pre-terminal one (about 61 seconds to classify 81,075 nodes).

	Section 2			Section 3			Section 24		
Nodes	pos	neg	tot	pos	neg	tot	pos	neg	tot
Internal	11,847	71,126	82,973	6,403	53,591	59,994	7,525	50,123	57,648
Pre-terminal	894	114,052	114,946	620	86,232	86,852	709	80,366	81,075
Both	12,741	185,178	197,919	7,023	139,823	146,846	8,234	130,489	138,723

Table 1: Tree nodes of the sentences from sections 2, 3 and 24 of the PropBank. pos and neg are the nodes that exactly cover arguments and all the other nodes, respectively.

Tagging strategy	Monolithic		Combined					
	CPU_{time}	F1	CPU_{time}			F1		
			Int	Pre	All	Int	Pre	All
PAF	5,179.18	75.24	1,794.92	56.72	1,851.64	79.93	79.39	79.89
MPAF	3,131.56	82.07	1,410.10	60.99	1,471.09	82.20	79.14	81.96

Table 2: F1 comparison between PAF and MPAF based kernels using different classification strategies. Int, Pre and ALL are the internal, pre-terminal and combined classifiers. The CPU time refers to the classification time in seconds of all Section 24.

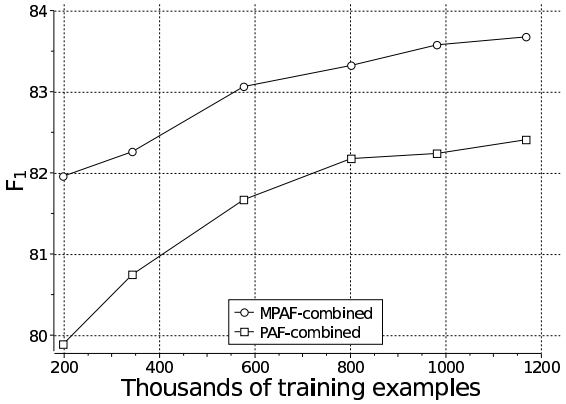


Figure 4: Learning curve comparison between the PAF and MPAF F1 measures using the combined classifier.

Third, the combined classifier approach seems quite feasible as its F_1 is almost equal to the monolithic one (81.96 vs. 82.07) in case of MPAF and even superior when using PAF (79.89 vs. 75.34). This result confirms the observation given in Section 3.1 about the importance of reducing the number of substructures common to PAFs associated with correct and incorrect boundaries.

Finally, we trained the combined boundary classifiers with sets of increasing size to derive the learning curves of the PAF and MPAF models. To have more significant results, we increased the training set by using also sections from 03 to 07. Figure 4 shows that the MPAF approach is constantly over the PAF. Consider also that the marking strategy has a lesser impact on the combined classifier.

4.3 Argument Classification Results

In these experiments we tested different kernels on the argument classification task. As some arguments have a very small number of training instances in a single section, we also used Section 03 for training and we continued to test on only Section 24.

The results of the multiclassifiers on 59 argument types[2] (e.g. constituted by 59 binary classifiers in the monolithic approach) are reported in Table 3. The rows from 3 to 5 report the accuracy when using the PAF, MPAF and CPAF whereas the rows from 6 to 8 show the accuracy for the complete argument structure approaches, i.e. MST, MMST and CMST.

More in detail, Column 2 shows the accuracy of the monolithic multi-argument classifiers whereas Columns 3, 4 and 5 report the accuracy of the internal, pre-terminal and combined multi-argument classifiers, respectively.

We note that:

First, the two classifier approach does not improve the monolithic approach accuracy. Indeed, the subtrees describing different argument types are quite different and this property holds also for the pre-terminal nodes. However, we still measured a remarkable improvement in efficiency.

Second, MPAF is the best kernel. This confirms the outcome on boundary detection experiments. The fact that it is more accurate than CPAF reveals that we need to distin-

[2] 7 for the core arguments (A0...AA), 13 for the adjunct arguments (AM-*), 19 for the argument references (R-*) and 20 for the continuations (C-*).

Tagging strategy	Monolithic	Combined		
		Internal nodes	Pre-terminals	Overall
PAF	75.06	74.16	85.61	75.15
MPAF	77.17	76.25	85.76	77.07
CPAF	76.79	75.68	85.76	76.54
MST	34.80	36.52	78.14	40.10
MMST	72.55	71.59	86.32	72.86
CMST	73.21	71.93	86.32	73.17

Table 3: Accuracy produced by different tree kernels on argument classification. We trained on sections 02 and 03 and tested on Section 24.

guish the argument node from the other nodes. To explain this, suppose that two argument nodes, $\texttt{NP}_1$ and $\texttt{NP}_2$, dominate the following structures: $\texttt{[NP}_1$ `[NP [DT NN]][PP]]` and $\texttt{[NP}_2$ `[DT NN]]`. If we mark only the argument node we obtain `[NP-B [NP [DT NN]][PP]]` and `[NP-B [DT NN]]` which have no structure in common. In contrast, if we mark them completely, i.e. `[NP-B [NP-B [DT-B NN-B]][PP-B]]` and `[NP-B [DT-B NN-B]]`, they will share the subtree `[NP-B [DT-B NN-B]]`. Thus, although it may seem counterintuitive, by marking only one node, we obtain more specific substructures. Of course, if we use different labels for the argument nodes and their descendants, we obtain the same specialization effect.

Finally, if we do not mark the target argument in the MSTs, we obtain a very low result (i.e. 40.10%) as expected. When we mark the covering node or the complete argument subtree we obtain an acceptable accuracy. Unfortunately, such accuracy is lower than the one produced by PAFs, e.g. 73.17% vs. 77.07%, thus it may seem that the additional information provided by the whole argument structure is not effective. A more careful analysis can be carried out by considering a CMST as composed by a PAF and the rest of the argument structure. We observe that some pieces of information provided by a PAF are not derivable by a CMST (or a MMST). For example, Figure 1 shows that the PAF contains the subtree `[VP [V NP]]` while the associated CMST (see Figure 2) contains `[VP [V NP PP]]`. The latter structure is larger and more sparse and consequently, the learning machine applied to CMSTs (or MMSTs) performs a more difficult generalization task. This problem is emphasized by our use of the adjuncts in the design of MSTs. As adjuncts tend to be the same for many predicates they do not provide a very discriminative information.

5 Discussions and Conclusions

The design of automatic systems for the labeling of semantic roles requires the solution of complex problems. Among others, feature engineering is made difficult by the structural nature of the data, i.e. features should represent information contained in automatic parse trees. This raises two problems: (1) the modeling of effective features, partially solved in the literature work and (2) the implementation of the software for the extraction of a large number of such features.

A system completely based on tree kernels alleviate both problems as (1) kernel functions automatically generate features and (2) only a procedure for subtree extraction is needed. Although some of the manual designed features seem to be superior to those derived with tree kernels, their combination seems still worth applying.

In this paper, we have improved tree kernels by studying different strategies: MPAF and the combined classifier (for internal and pre-terminal nodes) highly improve efficiency and accuracy in both the boundary detection and argument classification tasks. In particular, MPAF improves the old PAF-based tree kernel of about 8 absolute percent points in the boundary classification task, and when used along the combined classifier approach the speed of the model increases of 3.5 times. In case of argument classification the improvement is less evident but still consistent, about 2%.

We have also studied tree representations based on complete argument structures (MSTs). Our preliminary results seem to suggest that additional information extracted from other arguments is not effective. However, such findings are affected by two main problems: (1) We used adjuncts in the tree representation. They are likely to add more noise than useful information for the recognition of the argument type. (2) The traditional PAF contains subtrees that cannot be derived by the

MMSTs, thus we should combine these structures rather than substituting one with the other.

In the future, we plan to extend this study as follows:

First, our results are computed individually for boundary and classification tasks. Moreover, in our experiments, we removed arguments whose PAF or MST could not be extracted due to errors in parse trees. Thus, we provided only indicative accuracy to compare the different tree kernels. A final evaluation of the most promising structures using the CoNLL 2005 evaluator should be carried out to obtain a sound evaluation.

Second, as PAFs and MSTs should be combined to generate more information, we are going to carry out a set of experiments that combine different kernels associated with different subtrees. Moreover, as shown in (Basili and Moschitti, 2005; Moschitti, 2006), there are other tree kernel functions that generate different fragment types. The combination of such functions with the marking strategies may provide more general and effective kernels.

Third, once the final set of the most promising kernels is established, we would like to use all the available CoNLL 2005 data. This would allow us to study the potentiality of our approach by exactly comparing with literature work.

Next, our fast tree kernel function along with the combined classification approach and the improved tree representation make the learning and classification much faster so that the overall running time is comparable with polynomial kernels. However, when these latter are used with SVMs the running time is prohibitive when very large datasets (e.g. millions of instances) are targeted. Exploiting tree kernel derived features in a more efficient way is thus an interesting line of future research.

Finally, as CoNLL 2005 has shown that the most important contribution relates on re-ranking predicate argument structures based on one single tree (Toutanova et al., 2005) or several trees (Punyakanok et al., 2005), we would like to use tree kernels for the re-ranking task.

Acknowledgments

This research is partially supported by the European project, PrestoSpace (FP6-IST-507336).

References

Roberto Basili and Alessandro Moschitti. 2005. *Automatic Text Categorization: from Information Retrieval to Support Vector Learning*. Aracne Press, Rome, Italy.

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the CoNLL-2005 shared task: Semantic role labeling. In *Proceedings of CoNLL'05*.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the NACL'00*.

Michael Collins and Nigel Duffy. 2002. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *ACL'02*.

Aron Culotta and Jeffrey Sorensen. 2004. Dependency tree kernels for relation extraction. In *Proceedings of ACL'04*.

Chad Cumby and Dan Roth. 2003. Kernel methods for relational learning. In *Proceedings of ICML'03*.

Daniel Gildea and Daniel Jurasfky. 2002. Automatic labeling of semantic roles. *Computational Linguistic*, 28(3):496–530.

R. Jackendoff. 1990. *Semantic Structures, Current Studies in Linguistics series*. Cambridge, Massachusetts: The MIT Press.

T. Joachims. 1999. Making large-scale SVM learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*.

Christopher R. Johnson and Charles J. Fillmore. 2000. The framenet tagset for frame-semantic and syntactic coding of predicate-argument structure. In *In the Proceedings ANLP-NAACL*.

Paul Kingsbury and Martha Palmer. 2002. From Treebank to PropBank. In *Proceedings of LREC'02*.

M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. 1993. Building a large annotated corpus of english: The Penn Treebank. *Computational Linguistics*, 19:313–330.

Alessandro Moschitti. 2004. A study on convolution kernels for shallow semantic parsing. In *Proceedings of ACL'04*, Barcelona, Spain.

Alessandro Moschitti, Bonaventura Coppola, Daniele Pighin, and Roberto Basili. 2005. Engineering of syntactic features for shallow semantic parsing. In *of the ACL05 Workshop on Feature Engineering for Machine Learning in Natural Language Processing*, USA.

Alessandro Moschitti. 2006. Making tree kernels practical for natural language learning. In *Proceedings of EACL'06*, Trento, Italy.

Sameer Pradhan, Kadri Hacioglu, Valeri Krugler, Wayne Ward, James H. Martin, and Daniel Jurafsky. 2005. Support vector learning for semantic argument classification. *Machine Learning Journal*.

V. Punyakanok, D. Roth, and W. Yih. 2005. The necessity of syntactic parsing for semantic role labeling. In *Proceedings of IJCAI'05*.

Kristina Toutanova, Aria Haghighi, and Christopher Manning. 2005. Joint learning improves semantic role labeling. In *Proceedings of ACL'05*.

Nianwen Xue and Martha Palmer. 2004. Calibrating features for semantic role labeling. In *Proceedings of EMNLP 2004*.

D. Zelenko, C. Aone, and A. Richardella. 2003. Kernel methods for relation extraction. *Journal of Machine Learning Research*.

Syntagmatic Kernels: a Word Sense Disambiguation Case Study

Claudio Giuliano and **Alfio Gliozzo** and **Carlo Strapparava**
ITC-irst, Istituto per la Ricerca Scientifica e Tecnologica
I-38050, Trento, ITALY
{giuliano,gliozzo,strappa}@itc.it

Abstract

In this paper we present a family of kernel functions, named Syntagmatic Kernels, which can be used to model syntagmatic relations. Syntagmatic relations hold among words that are typically collocated in a sequential order, and thus they can be acquired by analyzing word sequences. In particular, Syntagmatic Kernels are defined by applying a Word Sequence Kernel to the local contexts of the words to be analyzed. In addition, this approach allows us to define a semi supervised learning schema where external lexical knowledge is plugged into the supervised learning process. Lexical knowledge is acquired from both unlabeled data and hand-made lexical resources, such as WordNet. We evaluated the syntagmatic kernel on two standard Word Sense Disambiguation tasks (i.e. English and Italian lexical-sample tasks of Senseval-3), where the syntagmatic information plays a crucial role. We compared the Syntagmatic Kernel with the standard approach, showing promising improvements in performance.

1 Introduction

In computational linguistics, it is usual to deal with sequences: words are sequences of letters and syntagmatic relations are established by sequences of words. Sequences are analyzed to measure morphological similarity, to detect multiwords, to represent syntagmatic relations, and so on. Hence modeling syntagmatic relations is crucial for a wide variety of NLP tasks, such as Named Entity Recognition (Gliozzo et al., 2005a) and Word Sense Disambiguation (WSD) (Strapparava et al., 2004).

In general, the strategy adopted to model syntagmatic relations is to provide bigrams and trigrams of collocated words as features to describe local contexts (Yarowsky, 1994), and each word is regarded as a different instance to classify. For instance, occurrences of a given class of named entities (such as *names of persons*) can be discriminated in texts by recognizing word patterns in their local contexts. For example the token *Rossi*, whenever is preceded by the token *Prof.*, often represents the name of a person. Another task that can benefit from modeling this kind of relations is WSD. To solve ambiguity it is necessary to analyze syntagmatic relations in the local context of the word to be disambiguated. In this paper we propose a kernel function that can be used to model such relations, the *Syntagmatic Kernel*, and we apply it to two (English and Italian) lexical-sample WSD tasks of the Senseval-3 competition (Mihalcea and Edmonds, 2004).

In a lexical-sample WSD task, training data are provided as a set of texts, in which for each text a given target word is manually annotated with a sense from a predetermined set of possibilities. To model syntagmatic relations, the typical supervised learning framework adopts as features bigrams and trigrams in a local context. The main drawback of this approach is that non contiguous or shifted col-

locations cannot be identified, decreasing the generalization power of the learning algorithm. For example, suppose that the verb *to score* has to be disambiguated into the sentence "Ronaldo *scored* the goal", and that the sense tagged example "the football player `scores#1` the first goal" is provided for training. A traditional feature mapping would extract the bigram $\mathbf{w_{+1}\_w_{+2}}$**:the_goal** to represent the former, and the bigram $\mathbf{w_{+1}\_w_{+2}}$**:the_first** to index the latter. Evidently such features will not match, leading the algorithm to a misclassification.

In the present paper we propose the Syntagmatic Kernel as an attempt to solve this problem. The Syntagmatic Kernel is based on a Gap-Weighted Subsequences Kernel (Shawe-Taylor and Cristianini, 2004). In the spirit of Kernel Methods, this kernel is able to compare sequences directly in the input space, avoiding any explicit feature mapping. To perform this operation, it counts how many times a (non-contiguous) subsequence of symbols u of length n occurs in the input string s, and penalizes non-contiguous occurrences according to the number of the contained gaps. To define our Syntagmatic Kernel, we adapted the generic definition of the Sequence Kernels to the problem of recognizing collocations in local word contexts.

In the above definition of Syntagmatic Kernel, only exact word-matches contribute to the similarity. One shortcoming of this approach is that (near-)synonyms will never be considered similar, leading to a very low generalization power of the learning algorithm, that requires a huge amount of data to converge to an accurate prediction. To solve this problem we provided external lexical knowledge to the supervised learning algorithm, in order to define a "soft-matching" schema for the kernel function. For example, if we consider as equivalent the terms *Ronaldo* and *football_player*, the proposition "*The football_player scored the first goal*" is equivalent to the sentence "*Ronaldo scored the first goal*", providing a strong evidence to disambiguate the latter occurrence of the verb.

We propose two alternative soft-matching criteria exploiting two different knowledge sources: (i) hand made resources and (ii) unsupervised term similarity measures. The first approach performs a soft-matching among all those synonyms words in WordNet, while the second exploits domain relations, acquired from unlabeled data, for the same purpose.

Our experiments, performed on two standard WSD benchmarks, show the superiority of the Syntagmatic Kernel with respect to a classical flat vector representation of bigrams and trigrams.

The paper is structured as follows. Section 2 introduces the Sequence Kernels. In Section 3 the Syntagmatic Kernel is defined. Section 4 explains how soft-matching can be exploited by the Collocation Kernel, describing two alternative criteria: WordNet Synonymy and Domain Proximity. Section 5 gives a brief sketch of the complete WSD system, composed by the combination of different kernels, dealing with syntagmatic and paradigmatic aspects. Section 6 evaluates the Syntagmatic Kernel, and finally Section 7 concludes the paper.

2 Sequence Kernels

The basic idea behind kernel methods is to embed the data into a suitable feature space $\mathcal{F}$ via a mapping function $\phi : \mathcal{X} \rightarrow \mathcal{F}$, and then use a linear algorithm for discovering nonlinear patterns. Instead of using the explicit mapping ϕ, we can use a kernel function $K : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$, that corresponds to the inner product in a feature space which is, in general, different from the input space.

Kernel methods allow us to build a modular system, as the kernel function acts as an interface between the data and the learning algorithm. Thus the kernel function becomes the only domain specific module of the system, while the learning algorithm is a general purpose component. Potentially any kernel function can work with any kernel-based algorithm. In our system we use Support Vector Machines (Cristianini and Shawe-Taylor, 2000).

Sequence Kernels (or *String Kernels*) are a family of kernel functions developed to compute the inner product among images of strings in high-dimensional feature space using dynamic programming techniques (Shawe-Taylor and Cristianini, 2004). The Gap-Weighted Subsequences Kernel is the most general Sequence Kernel. Roughly speaking, it compares two strings by means of the number of contiguous and non-contiguous substrings of a given length they have in common. Non contiguous occurrences are penalized according to the number of gaps they contain.

Formally, let Σ be an alphabet of $|\Sigma|$ symbols, and $s = s_1 s_2 \dots s_{|s|}$ a finite sequence over Σ (i.e. $s_i \in \Sigma, 1 \leqslant i \leqslant |s|$). Let $\mathbf{i} = [i_1, i_2, \dots, i_n]$, with $1 \leqslant i_1 < i_2 < \dots < i_n \leqslant |s|$, be a subset of the indices in s: we will denote as $s[\mathbf{i}] \in \Sigma^n$ the subsequence $s_{i_1} s_{i_2} \dots s_{i_n}$. Note that $s[\mathbf{i}]$ does not necessarily form a contiguous subsequence of s. For example, if s is the sequence "Ronaldo scored the goal" and $\mathbf{i} = [2, 4]$, then $s[\mathbf{i}]$ is "scored goal". The length spanned by $s[\mathbf{i}]$ in s is $l(\mathbf{i}) = i_n - i_1 + 1$. The feature space associated with the Gap-Weighted Subsequences Kernel of length n is indexed by $I = \Sigma^n$, with the embedding given by

$$\phi^n_u(s) = \sum_{\mathbf{i}:u=s[\mathbf{i}]} \lambda^{l(\mathbf{i})}, u \in \Sigma^n, \tag{1}$$

where $\lambda \in]0, 1]$ is the decay factor used to penalize non-contiguous subsequences[1]. The associate kernel is defined as

$$K_n(s,t) = \langle \phi^n(s), \phi^n(t) \rangle = \sum_{u \in \Sigma^n} \phi^n_u(s) \phi^n_u(t). \tag{2}$$

An explicit computation of Equation 2 is unfeasible even for small values of n. To evaluate more efficiently K_n, we use the recursive formulation proposed in (Lodhi et al., 2002; Saunders et al., 2002; Cancedda et al., 2003) based on a dynamic programming implementation. It is reported in the following equations:

$$K'_0(s,t) = 1, \forall s, t, \tag{3}$$

$$K'_i(s,t) = 0, \text{if } \min(|s|,|t|) < i, \tag{4}$$

$$K''_i(s,t) = 0, \text{if } \min(|s|,|t|) < i, \tag{5}$$

$$K''_i(sx,ty) = \begin{cases} \lambda K''_i(sx,t), & \text{if } x \neq y; \\ \lambda K''_i(sx,t) + \lambda^2 K'_{i-1}(s,t), & \text{otherwise.} \end{cases} \tag{6}$$

$$K'_i(sx,t) = \lambda K'_i(s,t) + K''_i(sx,t), \tag{7}$$

$$K_n(s,t) = 0, \text{if } \min(|s|,|t|) < n, \tag{8}$$

$$K_n(sx,t) = K_n(s,t) + \sum_{j:t_j=x} \lambda^2 K'_{n-1}(s, t[1:j-1]), \tag{9}$$

K'_n and K''_n are auxiliary functions with a similar definition as K_n used to facilitate the computation. Based on all definitions above, K_n can be computed in $O(n|s||t|)$. Using the above recursive definition, it turns out that computing all kernel values for subsequences of lengths up to n is not significantly more costly than computing the kernel for n only.

In the rest of the paper we will use the normalised version of the kernel (Equation 10) to keep the values comparable for different values of n and to be independent from the length of the sequences.

$$\hat{K}(s,t) = \frac{K(s,t)}{\sqrt{K(s,s)K(t,t)}}. \tag{10}$$

3 The Syntagmatic Kernel

As stated in Section 1, syntagmatic relations hold among words arranged in a particular temporal order, hence they can be modeled by Sequence Kernels. The Syntagmatic Kernel is defined as a linear combination of Gap-Weighted Subsequences Kernels that operate at word and PoS tag level. In particular, following the approach proposed by Cancedda et al. (2003), it is possible to adapt sequence kernels to operate at word level by instancing the alphabet Σ with the vocabulary $\mathcal{V} = \{w_1, w_2, \dots, w_k\}$. Moreover, we restricted the generic definition of the Gap-Weighted Subsequences Kernel to recognize collocations in the local context of a specified word. The resulting kernel, called *n-gram* Collocation Kernel (K^n_{Coll}), operates on sequences of lemmata around a specified word l_0 (i.e. $l_{-3}, l_{-2}, l_{-1}, l_0, l_{+1}, l_{+2}, l_{+3}$). This formulation allows us to estimate the number of common (sparse) subsequences of lemmata (i.e. collocations) between two examples, in order to capture syntagmatic similarity.

Analogously, we defined the PoS Kernel (K^n_{PoS}) to operate on sequences of PoS tags p_{-3}, p_{-2}, p_{-1}, p_0, p_{+1}, p_{+2}, p_{+3}, where p_0 is the PoS tag of l_0.

The Collocation Kernel and the PoS Kernel are defined by Equations 11 and 12, respectively.

$$K_{Coll}(s,t) = \sum_{l=1}^{n} K^l_{Coll}(s,t) \tag{11}$$

and

$$K_{PoS}(s,t) = \sum_{l=1}^{n} K^l_{PoS}(s,t). \tag{12}$$

Both kernels depend on the parameter n, the length of the non-contiguous subsequences, and λ, the de-

[1] Notice that by choosing $\lambda = 1$ sparse subsequences are not penalized. On the other hand, the kernel does not take into account sparse subsequences with $\lambda \to 0$.

cay factor. For example, K^2_{Coll} allows us to represent all (sparse) bi-grams in the local context of a word.

Finally, the Syntagmatic Kernel is defined as

$$K_{Synt}(s,t) = K_{Coll}(s,t) + K_{PoS}(s,t). \quad (13)$$

We will show that in WSD, the Syntagmatic Kernel is more effective than standard bigrams and trigrams of lemmata and PoS tags typically used as features.

4 Soft-Matching Criteria

In the definition of the Syntagmatic Kernel only exact word matches contribute to the similarity. To overcome this problem, we further extended the definition of the Gap-Weigthed Subsequences Kernel given in Section 2 to allow soft-matching between words. In order to develop soft-matching criteria, we follow the idea that two words can be substituted preserving the meaning of the whole sentence if they are paradigmatically related (e.g. synomyns, hyponyms or domain related words). If the meaning of the proposition as a whole is preserved, the meaning of the lexical constituents of the sentence will necessarily remain unchanged too, providing a viable criterion to define a soft-matching schema. This can be implemented by "plugging" external paradigmatic information into the Collocation kernel.

Following the approach proposed by (Shawe-Taylor and Cristianini, 2004), the soft-matching Gap-Weighted Subsequences Kernel is now calculated recursively using Equations 3 to 5, 7 and 8, replacing Equation 6 by the equation:

$$K''_i(sx,ty) = \lambda K''_i(sx,t) + \lambda^2 a_{xy} K'_{i-1}(s,t), \forall x,y, \quad (14)$$

and modifying Equation 9 to:

$$K_n(sx,t) = K_n(s,t) + \sum_j^{|t|} \lambda^2 a_{xt_j} K'_{n-1}(s, t[1:j-1]). \quad (15)$$

where a_{xy} are entries in a similarity matrix $\mathbf{A}$ between symbols (words). In order to ensure that the resulting kernel is valid, $\mathbf{A}$ must be positive semi-definite.

In the following subsections, we describe two alternative soft-matching criteria based on WordNet Synonymy and Domain Proximity. In both cases, to show that the similarity matrices are a positive semi-definite we use the following result:

Proposition 1 *A matrix* $\mathbf{A}$ *is positive semi-definite if and only if* $\mathbf{A} = \mathbf{B}^T\mathbf{B}$ *for some real matrix* $\mathbf{B}$.

The proof is given in (Shawe-Taylor and Cristianini, 2004).

4.1 WordNet Synonymy

The first solution we have experimented exploits a lexical resource representing paradigmatic relations among terms, i.e. WordNet. In particular, we used WordNet-1.7.1 for English and the Italian part of MultiWordNet[2].

In order to find a similarity matrix between terms, we defined a vector space where terms are represented by the WordNet synsets in which such terms appear. Hence, we can view a term as vector in which each dimension is associated with one synset. The term-by-synset matrix $\mathbf{S}$ is then the matrix whose rows are indexed by the synsets. The entry x_{ij} of $\mathbf{S}$ is 1 if the synset s_j contains the term w_i, and 0 otherwise. The term-by-synset matrix $\mathbf{S}$ gives rise to the similarity matrix $\mathbf{A} = \mathbf{S}\mathbf{S}^T$ between terms. Since $\mathbf{A}$ can be rewritten as $\mathbf{A} = (\mathbf{S}^T)^T\mathbf{S}^T = \mathbf{B}^T\mathbf{B}$, it follows directly by Proposition 1 that it is positive semi-definite.

It is straightforward to extend the soft-matching criterion to include hyponym relation, but we achieved worse results. In the evaluation section we will not report such results.

4.2 Domain Proximity

The approach described above requires a large scale lexical resource. Unfortunately, for many languages, such a resource is not available. Another possibility for implementing soft-matching is introducing the notion of Semantic Domains.

Semantic Domains are groups of strongly paradigmatically related words, and can be acquired automatically from corpora in a totally unsupervised way (Gliozzo, 2005). Our proposal is to exploit a Domain Proximity relation to define a soft-matching criterion on the basis of an unsupervised similarity metric defined in a Domain Space. The Domain Space can be determined once a Domain

[2] http://multiwordnet.itc.it

Model (DM) is available. This solution is evidently cheaper, because large collections of unlabeled texts can be easily found for every language.

A DM is represented by a $k \times k'$ rectangular matrix $\mathbf{D}$, containing the domain relevance for each term with respect to each domain, as illustrated in Table 1. DMs can be acquired from texts by exploiting a lexical coherence assumption (Gliozzo, 2005). To this aim, Term Clustering algorithms can be used: a different domain is defined for each cluster, and the degree of association between terms and clusters, estimated by the unsupervised learning algorithm, provides a domain relevance function. As a clustering technique we exploit Latent Semantic Analysis (LSA), following the methodology described in (Gliozzo et al., 2005b). This operation is done offline, and can be efficiently performed on large corpora.

	MEDICINE	COMPUTER_SCIENCE
HIV	1	0
AIDS	1	0
virus	0.5	0.5
laptop	0	1

Table 1: Example of Domain Model.

LSA is performed by means of SVD of the term-by-document matrix $\mathbf{T}$ representing the corpus. The SVD algorithm can be exploited to acquire a domain matrix $\mathbf{D}$ from a large corpus in a totally unsupervised way. SVD decomposes the term-by-document matrix $\mathbf{T}$ into three matrices $\mathbf{T} = \mathbf{V}\mathbf{\Sigma_k}\mathbf{U}^T$ where $\mathbf{\Sigma_k}$ is the diagonal $k \times k$ matrix containing the k singular values of $\mathbf{T}$. $\mathbf{D} = \mathbf{V}\mathbf{\Sigma_{k'}}$ where $k' \ll k$.

Once a DM has been defined by the matrix $\mathbf{D}$, the Domain Space is a k' dimensional space, in which both texts and terms are represented by means of Domain Vectors (DVs), i.e. vectors representing the domain relevances among the linguistic object and each domain. The DV $\vec{w_i'}$ for the term $w_i \in \mathcal{V}$ is the i^{th} row of $\mathbf{D}$, where $\mathcal{V} = \{w_1, w_2, \ldots, w_k\}$ is the vocabulary of the corpus.

The term-by-domain matrix $\mathbf{D}$ gives rise to the term-by-term similarity matrix $\mathbf{A} = \mathbf{D}\mathbf{D}^T$ among terms. It follows from Proposition 1 that $\mathbf{A}$ is positive semi-definite.

5 Kernel Combination for WSD

To improve the performance of a WSD system, it is possible to combine different kernels. Indeed, we followed this approach in the participation to Senseval-3 competition, reaching the state-of-the-art in many lexical-sample tasks (Strapparava et al., 2004). While this paper is focused on Syntagmatic Kernels, in this section we would like to spend some words on another important component for a complete WSD system: the Domain Kernel, used to model domain relations.

Syntagmatic information alone is not sufficient to define a full kernel for WSD. In fact, in (Magnini et al., 2002), it has been claimed that knowing the domain of the text in which the word is located is a crucial information for WSD. For example the (domain) polysemy among the COMPUTER_SCIENCE and the MEDICINE senses of the word `virus` can be solved by simply considering the domain of the context in which it is located.

This fundamental aspect of lexical polysemy can be modeled by defining a kernel function to estimate the domain similarity among the contexts of the words to be disambiguated, namely the *Domain Kernel*. The Domain Kernel measures the similarity among the topics (domains) of two texts, so to capture domain aspects of sense distinction. It is a variation of the Latent Semantic Kernel (Shawe-Taylor and Cristianini, 2004), in which a DM is exploited to define an explicit mapping $\mathcal{D} : \mathbb{R}^k \rightarrow \mathbb{R}^{k'}$ from the Vector Space Model (Salton and McGill, 1983) into the Domain Space (see Section 4), defined by the following mapping:

$$\mathcal{D}(\vec{t_j}) = \vec{t_j}(\mathbf{I^{IDF}}\mathbf{D}) = \vec{t_j'} \tag{16}$$

where $\mathbf{I^{IDF}}$ is a $k \times k$ diagonal matrix such that $i_{i,i}^{IDF} = IDF(w_i)$, $\vec{t_j}$ is represented as a row vector, and $IDF(w_i)$ is the *Inverse Document Frequency* of w_i. The Domain Kernel is then defined by:

$$K_D(t_i, t_j) = \frac{\langle \mathcal{D}(t_i), \mathcal{D}(t_j) \rangle}{\sqrt{\langle \mathcal{D}(t_j), \mathcal{D}(t_j) \rangle \langle \mathcal{D}(t_i), \mathcal{D}(t_i) \rangle}} \tag{17}$$

The final system for WSD results from a combination of kernels that deal with syntagmatic and paradigmatic aspects (i.e. PoS, collocations, bag of words, domains), according to the following kernel

combination schema:

$$K_C(x_i, x_j) = \sum_{l=1}^{n} \frac{K_l(x_i, x_j)}{\sqrt{K_l(x_j, x_j) K_l(x_i, x_i)}} \quad (18)$$

6 Evaluation

In this section we evaluate the Syntagmatic Kernel, showing that it improves over the standard feature extraction technique based on bigrams and trigrams of words and PoS tags.

6.1 Experimental settings

We conducted the experiments on two lexical sample tasks (English and Italian) of the Senseval-3 competition (Mihalcea and Edmonds, 2004). In lexical-sample WSD, after selecting some target words, training data is provided as a set of texts. For each text a given target word is manually annotated with a sense from a predetermined set of possibilities. Table 2 describes the tasks by reporting the number of words to be disambiguated, the mean polysemy, and the dimension of training, test and unlabeled corpora. Note that the organizers of the English task did not provide any unlabeled material. So for English we used a domain model built from the training partition of the task (obviously skipping the sense annotation), while for Italian we acquired the DM from the unlabeled corpus made available by the organizers.

	#w	*pol*	*# train*	*# test*	*# unlab*
English	57	6.47	7860	3944	7860
Italian	45	6.30	5145	2439	74788

Table 2: Dataset descriptions.

6.2 Performance of the Syntagmatic Kernel

Table 3 shows the performance of the Syntagmatic Kernel on both data sets. As baseline, we report the result of a standard approach consisting on explicit bigrams and trigrams of words and PoS tags around the words to be disambiguated (Yarowsky, 1994). The results show that the Syntagmatic Kernel outperforms the baseline in any configuration (hard/soft-matching). The soft-matching criteria further improve the classification performance. It is interesting to note that the Domain Proximity methodology obtained better results than WordNet Synonymy. The different results observed between Italian and English using the Domain Proximity soft-matching criterion are probably due to the small size of the unlabeled English corpus.

Standard approach		
	English	*Italian*
Bigrams and trigrams	67.3	51.0
Syntagmatic Kernel		
Hard matching	67.7	51.9
Soft matching (WordNet)	67.3	51.3
Soft matching (Domain proximity)	68.5	54.0

Table 3: Performance (F1) of the Syntagmatic Kernel.

In these experiments, the parameters n and λ are optimized by cross-validation. For K^n_{Coll}, we obtained the best results with $n = 2$ and $\lambda = 0.5$. For K^n_{PoS}, $n = 3$ and $\lambda \rightarrow 0$. The domain cardinality k' was set to 50.

Finally, the global performance (F1) of the full WSD system (see Section 5) on English and Italian lexical sample tasks is 73.3 for English and 61.3 for Italian. To our knowledge, these figures represent the current state-of-the-art on these tasks.

7 Conclusion and Future Work

In this paper we presented the Syntagmatic Kernels, i.e. a set of kernel functions that can be used to model syntagmatic relations for a wide variety of Natural Language Processing tasks. In addition, we proposed two soft-matching criteria for the sequence analysis, which can be easily modeled by relaxing the constraints in a Gap-Weighted Subsequences Kernel applied to local contexts of the word to be analyzed. Experiments, performed on two lexical sample Word Sense Disambiguation benchmarks, show that our approach further improves the standard techniques usually adopted to deal with syntagmatic relations. In addition, the Domain Proximity soft-matching criterion allows us to define a semi-supervised learning schema, improving the overall results.

For the future, we plan to exploit the Syntagmatic Kernel for a wide variety of Natural Language Processing tasks, such as Entity Recognition and Relation Extraction. In addition we are applying the soft matching criteria here defined to Tree Kernels,

in order to take into account lexical variability in parse trees. Finally, we are going to further improve the soft-matching criteria here proposed by exploring the use of entailment criteria for substitutability.

Acknowledgments

The authors were partially supported by the Onto-Text Project, funded by the Autonomous Province of Trento under the FUP-2004 research program.

References

N. Cancedda, E. Gaussier, C. Goutte, and J.M. Renders. 2003. Word-sequence kernels. *Journal of Machine Learning Research*, 32(6):1059–1082.

N. Cristianini and J. Shawe-Taylor. 2000. *An introduction to Support Vector Machines*. Cambridge University Press.

A. Gliozzo, C. Giuliano, and R. Rinaldi. 2005a. Instance filtering for entity recognition. *ACM SIGKDD Explorations, special Issue on Natural Language Processing and Text Mining*, 7(1):11–18, June.

A. Gliozzo, C. Giuliano, and C. Strapparava. 2005b. Domain kernels for word sense disambiguation. In *Proceedings of the 43rd annual meeting of the Association for Computational Linguistics (ACL-05)*, pages 403–410, Ann Arbor, Michigan, June.

A. Gliozzo. 2005. *Semantic Domains in Computational Linguistics*. Ph.D. thesis, ITC-irst/University of Trento.

H. Lodhi, J. Shawe-Taylor, N. Cristianini, and C. Watkins. 2002. Text classification using string kernels. *Journal of Machine Learning Research*, 2(3):419–444.

B. Magnini, C. Strapparava, G. Pezzulo, and A. Gliozzo. 2002. The role of domain information in word sense disambiguation. *Natural Language Engineering*, 8(4):359–373.

R. Mihalcea and P. Edmonds, editors. 2004. *Proceedings of SENSEVAL-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, Barcelona, Spain, July.

G. Salton and M.H. McGill. 1983. *Introduction to modern information retrieval*. McGraw-Hill, New York.

C. Saunders, H. Tschach, and J. Shawe-Taylor. 2002. Syllables and other string kernel extensions. In *Proceedings of 19th International Conference on Machine Learning (ICML02)*.

J. Shawe-Taylor and N. Cristianini. 2004. *Kernel Methods for Pattern Analysis*. Cambridge University Press.

C. Strapparava, C. Giuliano, and A. Gliozzo. 2004. Pattern abstraction and term similarity for word sense disambiguation: IRST at Senseval-3. In *Proceedings of SENSEVAL-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, Barcelona, Spain, July.

D. Yarowsky. 1994. Decision lists for lexical ambiguity resolution: Application to accent restoration in spanish and french. In *Proceedings of the 32nd Annual Meeting of the ACL*, pages 88–95, Las Cruces, New Mexico.

Learning to Identify Definitions using Syntactic Features

Ismail Fahmi and **Gosse Bouma**
Information Science
Groningen University
{i.fahmi,g.bouma}@rug.nl

Abstract

This paper describes an approach to learning concept definitions which operates on fully parsed text. A subcorpus of the Dutch version of Wikipedia was searched for sentences which have the syntactic properties of definitions. Next, we experimented with various text classification techniques to distinguish actual definitions from other sentences. A maximum entropy classifier which incorporates features referring to the position of the sentence in the document as well as various syntactic features, gives the best results.

1 Introduction

Answering definition questions is a challenge for question answering systems. Much work in QA has focused on answering factoid questions, which are characterized by the fact that given the question, one can typically make strong predictions about the type of expected answer (i.e. a date, name of a person, amount, etc.). Definition questions require a different approach, as a definition can be a phrase or sentence for which only very global characteristics hold.

In the CLEF 2005 QA task, 60 out of 200 questions were asking for the definition of a named entity (a person or organization) such as *Who is Goodwill Zwelithini?* or *What is IKEA?* Answers are phrases such as *current king of the Zulu nation*, or *Swedish home furnishings retailer*. For answering definition questions restricted to named entities, it generally suffices to search for noun phrases consisting of the named entity and a preceding or following nominal phrase. Bouma et al. (2005) extract all such noun phrases from the Dutch CLEF corpus off-line, and return the most frequent heads of co-occurring nominal phrases expanded with adjectival or prepositional modifiers as answer to named entity definition questions. The resulting system answers 50% of the CLEF 2005 definition questions correctly.

For a Dutch medical QA system, which is being developed as part of the IMIX project[1], several sets of test questions were collected. Approximately 15% of the questions are definition questions, such as *What is a runner's knee?* and *What is cerebrovascular accident?*. Answers to such questions (asking for the definition of a concept) are typically found in sentences such as *A runner's knee is a degenerative condition of the cartilage surface of the back of the knee cap, or patella* or *A cerebrovascular accident is a decrease in the number of circulating white blood cells (leukocytes) in the blood*. One approach to finding answers to concept definitions simply searches the corpus for sentences consisting of a subject, a copular verb, and a predicative phrase. If the concept matches the subject, the predicative phrase can be returned as answer. A preliminary evaluation of this technique in Tjong Kim Sang et al. (2005) revealed that only 18% of the extracted sentences (from a corpus consisting of a mixture of encyclopedic texts and web documents) is actually a definition. For instance, sentences such as *RSI is a major problem in the Netherlands*, *every suicide attempt is an emergency* or *an infection of the lungs is the most serious complication* are of the relevant syntactic form, but do not constitute definitions.

In this paper, we concentrate on a method for improving the precision of recognizing definition sentences. In particular, we investigate to what

[1] www.let.rug.nl/~gosse/Imix

extent machine learning techniques can be used to distinguish definitions from non-definitions in a corpus of sentences containing a subject, copular verb, and predicative phrase. A manually annotated subsection of the corpus was divided into definition and non-definition sentences. Next, we trained various classifiers using unigram and bigram features, and various syntactic features. The best classifier achieves a 60% error reduction compared to our baseline system.

2 Previous work

Work on identifying definitions from free text initially relied on manually crafted patterns without applying any machine learning technique. Klavans and Muresan (2000) set up a pattern extractor for their Definder system using a tagger and a finite state grammar. Joho and Sanderson (2000) retrieve descriptive phrases (*dp*) of query nouns (*qn*) from text to answer definition questions like *Who is qn?* Patterns such as '*dp* especially *qn*', as utilized by Hearst (1992), are used to extract names and their descriptions.

Similar patterns are also applied by Liu et al. (2003) to mine definitions of topic-specific concepts on the Web. As an additional assumption, specific documents dedicated to the concepts can be identified if they have particular HTML and hyperlink structures.

Hildebrandt et al. (2004) exploit surface patterns to extract as many relevant "nuggets" of information of a concept as possible. Similar to our work, a copular pattern NP_1 *be* NP_2 is used as one of the extraction patterns. Nuggets which do not begin with a determiner are discarded to filter out spurious nuggets (e.g., progressive tense). Nuggets extracted from every article in a corpus are then stored in a relational database. In the end, answering definition questions becomes as simple as looking up relevant terms from the database. This strategy is similar to our approach for answering definition questions.

The use of machine learning techniques can be found in Miliaraki and Androutsopoulos (2004) and Androutsopoulos and Galanis (2005) They use similar patterns as (Joho and Sanderson, 2000) to construct training attributes. Sager and L'Homme (1994) note that the definition of a term should at least always contain *genus* (term's category) and *species* (term's properties). Blair-Goldensohn et al. (2004) uses machine learning and manually crafted lexico-syntactic patterns to match sentences containing both a genus and species phrase for a given term.

There is an intuition that most of definition sentences are located at the beginning of documents. This lead to the use of sentence number as a good indicator of potential definition sentences. Joho and Sanderson (2000) use the position of the sentences as one of their ranking criteria, while Miliaraki and Androutsopoulos (2004), Androutsopoulos and Galanis (2005) and Blair-Goldensohn et al. (2004) apply it as one of their learning attributes.

3 Syntactic properties of potential definition sentences

To answer medical definition sentences, we used the medical pages of Dutch Wikipedia[2] as source. Medical pages were selected by selecting all pages mentioned on the *Healthcare* index page, and recursively including pages mentioned on retrieved pages as well.

The corpus was parsed syntactically by Alpino, a robust wide-coverage parser for Dutch (Malouf and van Noord, 2004). The result of parsing (illustrated in Figure 1) is a dependency graph. The Alpino-parser comes with an integrated named entity classifier which assigns distinct part-of-speech tags to person, organization, and geographical named entities.

Potential definition sentences are sentences containing a form of the verb *zijn*[3] (*to be*) with a subject and nominal predicative phrase as sisters. The syntactic pattern does not match sentences in which *zijn* is used as a possessive pronoun (*his*) and sentences where a form of *zijn* is used as an auxiliary. In the latter case, no predicative phrase complement will be found. On the other hand, we do include sentences in which the predicative phrase precedes the subject, as in *Onderdeel van de testis is de Leydig-cel* (*the Leydig cel is part of the testis*). As word order in Dutch is less strict than in English, it becomes relevant to include such non-canonical word orders as well.

A number of non-definition sentences that will be extracted using this method can be filtered by simple lexical methods. For instance, if the subject is headed by (the Dutch equivalents of) *cause, con-*

[2] nl.wikipedia.org

[3] Note that the example uses *ben* (the first person singular form of the verb) as root for *zijn*.

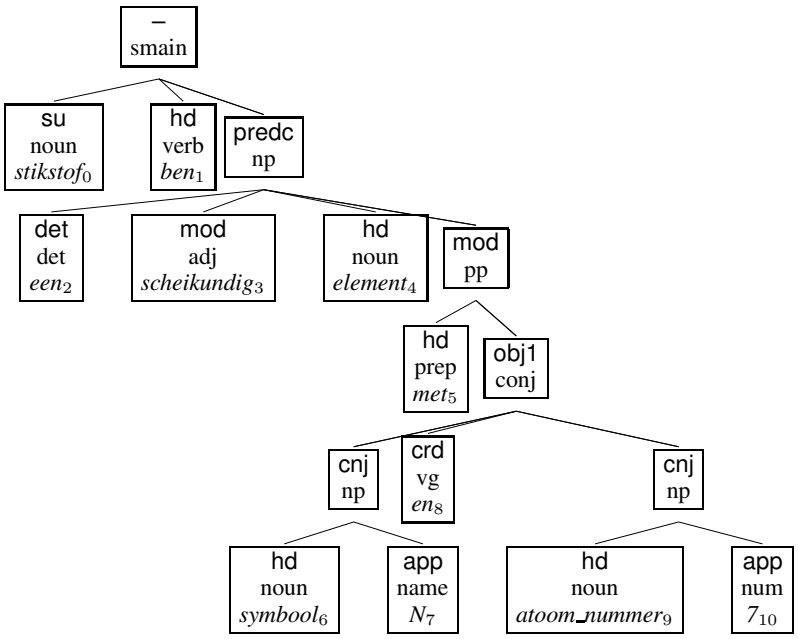


Figure 1: Parse of (the Dutch equivalent of) *Nitrogen is a chemical element with symbol N and atomic number 7*. Nodes are labelled with depedency relations and categories or part-of-speech tags, root forms, and string positions.

sequence, example, problem, result, feature, possibility, symptom, sign, etc., or contains the determiner *geen* (*no*), the sentence will not be included in the list of potential definitions.

However, even after applying the lexical filter, not all extracted sentences are definitions. In the next sections, we describe experiments aimed at increasing the accuracy of the extraction method.

4 Annotating training examples

To create evaluation and training data, 2500 extracted sentences were manually annotated as definition, non-definition, or undecided. One of the criteria for undecided sentences is that it mentions a characteristic of a definition but is not really a (complete) definition, for example, *Benzeen is carcinogeen* (*Benzene is a carcinogen*). The result of this annotation is given in Table 1. The annotated data was used both to evaluate the accuracy of the syntactic extraction method, and to training and evaluate material for the machine learning experiments as discussed in the next sections.

After discarding the undecided sentences, we are left with 2299 sentences, 1366 of which are definitions. This means that the accuracy of the extraction method using only syntax was 59%.[4] If we take sentence postion into account as well, and classify all first sentences as definitions and all other sentences as non-definitions, a baseline accuracy of 75,9% is obtained.

It is obvious from Table 1 that the first sentences of Wikipedia lemmas that match the syntactic pattern are almost always definitions. It seems that e.g. Google's[5] *define* query feature, when restricted to Dutch at least, relies heavily on this fact to answer definition queries. However it is also obvious that definition sentences can also be found in other positions. For documents from other sources, which are not as structured as Wikipedia, the first position sentence is likely to be an even weaker predictor of definition vs. non-definition sentences.

5 Attributes of definition sentences

We aim at finding the best attributes for classifying definition sentences. We experimented with combinations of the following attributes:

Text properties: bag-of-words, bigrams, and root forms. Punctuation is included as Klavans and Muresan (2000) observe that it can be used to recognize definitions (i.e. definitions tend to con-

[4]This is considerably higher than the estimated accuracy of 18% reported in Tjong Kim Sang et al. (2005). This is probably partly due to the fact that the current corpus consists of encyclopedic material only, whereas the corpus used in Tjong Kim Sang et al. (2005) contained web material from various sources, such as patient discussion groups, as well. The latter tends to contain more subjective and context-dependent material.

[5]`google.com`

Sentence position	Def	Non-def	Undecided
first	831	18	31
other	535	915	170
Total	1366	933	201

Table 1: Number of sentences in the first and other position of documents annotated as definition, non-definition, and undecided.

tain parentheses more often than non-definitions). No stopword filtering is applied as in our experiments it consistently decreased accuracy. Note that we include all bigrams in a sentence as feature. A different use of n-grams has been explored by Androutsopoulos and Galanis (2005) who add only n-grams ($n \in \{1,2,3\}$) occurring frequently either directly before or after a target term.

Document property: the position of each sentence in the document. This attribute has been frequently used in previous work and is motivated by the observation that definitions are likely to be located in the beginning of a document.

Syntactic properties: position of each subject in the sentence (*initial*, e.g. *X is Y*; or *non-initial*, e.g. *Y is X*), and of each subject and predicative complement: type of determiner (definite, indefinite, other). These attributes have not been investigated in previous work. In our experiments, sentence-initial subjects appear in 92% of the definition sentences and and 76% of the non-definition sentences. These values show that a definition sentence with a copular pattern tends to put its subject in the beginning. Two other attributes are used to encode the type of determiner of the subject and predicative compelement. As shown in Table 2, the majority of subjects in definition sentences have no determiner (62%), e.g. *Paracetamol is een pijnstillend en koortsverlagend middel* (*Paracetamol is an pain alleviating and a fever reducing medicine*), while in non-definition sentences subject determiners tend to be definite (50%), e.g. *De werkzame stof is acetylsalicylzuur* (*The operative substance is acetylsalicylacid*). Predicative complements, as shown in Table 3, tend to contain indefinite determiners in definition sentences (64%), e.g. *een pijnstillend . . . medicijn* (*a pain alleviating. . . medicine*), while in non-definition the determiner tends to be definite (33%), e.g. *Een fenomeen is de Landsgemeinde* (*A phenomenon is the Landsgemeinde*).

Type	Definition	Non-def
definite	23	**50**
indefinite	13	12
nodeterminer	**62**	29
other	2	9

Table 2: Percentage of determiner types of subjects in definition and non-definition sentences.

Type	Definition	Non-def
definite	23	**33**
indefinite	**64**	29
nodeterminer	8	1
other	4	28

Table 3: Percentage of determiner types of predicative complements in definition and non-definition sentences.

Named entity tags: named entity class (NEC) of subjects, e.g. location, person, organization, or no-class. A significant difference in the distribution of this feature between definition and non-definition sentences can be observed in Table 4. More definition sentences have named entity classes contained in their subjects (40.63%) compared to non-definition sentences (11.58%). We also experimented with named entity classes contained in predicative complements but it turned out that very few predicates contained named entities, and thus no significant differences in distribution between definition and non-definition sentences could be observed.

Features for lexical patterns, as used in (Androutsopoulos and Galanis, 2005), e.g. *qn which (is|was|are|were) dp*, are not added because in this experiment we investigate only a copular pattern. WordNet-based attributes are also excluded, given that coverage for Dutch (using EuroWordNet) tends to be less good than for English, and even for English their contribution is sometimes insignificant (Miliaraki and Androutsopoulos, 2004).

Type	Definition	Non-def
no-nec	59	88
location	10	4
organization	8	3
person	22	4

Table 4: Percentage of named-entity classes of subjects in definition and non-definition sentences.

word bigrams only	bigram + synt + pos
is a	first_sent
a	other_sent
are	is a
is	indef_pred
) is	no_det_subj
the	init_subj
is DIGITS	a
are the	are
this	is
or	other_det_pred
is of	) is
this/these	noninit_subj
atomic_number	def_subj
atomic_number DIGITS	the
with symbol	is DIGITS
and atomic_number	are the
that	this
chemical	or
a chemical	other_det_subj
chemical element	is of

Table 5: 20 most informative features for the systems using word bigrams only and word bigrams in combination with syntactic and sentence position features (word features have been translated into English).

We use the text classification tool Rainbow[6] (McCallum, 2000) to perform most of our experiments. Each sentence is represented as a string of words, possibly followed by bigrams, root forms, (combinations of) syntactic features, etc.

All experiments were performed by selecting only the 2000 highest ranked features according to information gain. In the experiments which include syntactic features, the most informative features tend to contain a fair number of syntactic features. This is illustrated for the configuration using bigrams, sentence position, and syntax in table 5. It supports our intuition that the position of subjects and the type of determiner of subjects and predicative complements are clues to recognizing definition sentences.

To investigate the effect of each attribute, we set up several configurations of training examples as described in Table 6. We start with using only bag-of-words or bigrams, and then combine them with other attribute sets.

[6] www.cs.cmu.edu/~mccallum/bow/rainbow/

Cfg	Description
1	using only bag-of-words
2	using only bigrams
3	combining bigrams & bag-of-words
4	adding syntactic properties to config. 3
5	adding syntactic properties & NEC to config. 3
6	adding sentence position to config. 3
7	adding root forms to config. 3
8	adding syntactic properties & sentence position to config. 3
9	adding syntactic properties, sentence position & NEC to config. 3
10	adding syntactic properties, sentence position & root forms to config. 3)
11	using all attributes (adding NEC to configuration 10)

Table 6: The description of the attribute configurations.

6 Learning-based methods

We apply three supervised learning methods to each of the attribute configurations in Table 6, namely naive Bayes, maximum entropy, and support vector machines (SVMs). Naive Bayes is a fast and easy to use classifier based on the probabilistic model of text and has often been used in text classification tasks as a baseline. Maximum entropy is a general estimation technique that has been used in many fields such as information retrieval and machine learning. Some experiments in text classification show that maximum entropy often outperforms naive Bayes, e.g. on two of three data sets in Nigam et al. (1999). SVMs are a new learning method but have been reported by Joachims (1998) to be well suited for learning in text classification.

We experiment with three kernel types of SVMs: linear, polynomial, and radial base function (RBF). Rainbow (McCallum, 2000) is used to examine these learning methods, except the RBF kernel for which libsvm (Chang and Lin, 2001) is used. Miliaraki and Androutsopoulos (2004) use a SVM with simple inner product (polynomial of first degree) kernel because higher degree polynomial kernels were reported as giving no improvement. However we want to experiment with

Cfg	NB	ME	svm1[a]	svm2[b]	svm3[c]
1	85.75 ± 0.57	85.35 ± 0.77	77.65 ± 0.87	78.39 ± 0.67	81.95 ± 0.82
2	87.77 ± 0.51	88.65 ± 0.54	84.02 ± 0.47	84.26 ± 0.52	85.38 ± 0.77
3	89.82 ± 0.53	88.82 ± 0.66	83.93 ± 0.57	84.24 ± 0.54	87.04 ± 0.95
4	85.22 ± 0.35	89.08 ± 0.50	84.93 ± 0.57	85.57 ± 0.53	87.77 ± 0.89
5	85.44 ± 0.45	91.38 ± 0.42	86.90 ± 0.48	86.90 ± 0.53	87.60 ± 0.87
6	**90.26 ± 0.71**	90.70 ± 0.48	85.26 ± 0.56	86.05 ± 0.64	88.52 ± 0.92
7	88.60 ± 0.81	88.99 ± 0.51	83.38 ± 0.38	84.69 ± 0.43	87.08 ± 0.87
8	86.40 ± 0.51	**92.21 ± 0.27**	86.57 ± 0.42	87.29 ± 0.47	88.77 ± 0.77
9	87.12 ± 0.52	90.83 ± 0.43	87.21 ± 0.42	**87.99 ± 0.53**	89.04 ± 0.67
10	87.60 ± 0.38	91.16 ± 0.43	86.68 ± 0.40	86.97 ± 0.41	88.91 ± 0.68
11	86.72 ± 0.46	91.16 ± 0.35	**87.47 ± 0.40**	87.05 ± 0.63	**89.47 ± 0.67**

[a] SVM with linear kernel (Rainbow)
[b] SVM with polynomial kernel (Rainbow)
[c] SVM with RBF kernel (libsvm)

Table 7: Accuracy and standard error (%) estimates for the dataset using naive Bayes (NB), maximum entropy (ME), and three SVM settings at the different attribute configurations.

the RBF (gaussian) kernel by selecting model parameters C (penalty for misclassification) and γ (function of the deviation of the Gaussian Kernel) so that the classifier can accurately predict testing data. This experiment is based on the argument that if a complete model selection using the gaussian kernel has been conducted, there is no need to consider linear SVM, because the RBF kernel with certain parameters (C, γ) has the same performance as the linear kernel with a penalty parameter $\tilde{C}$ (Keerthi and Lin, 2003).

Given the finite dataset, we use k-fold cross-validation (k = 20) to estimate the future performance of each classifier induced by its learning method and dataset. This estimation method introduces lower bias compared to a bootstrap method which has extremely large bias on some problems (Kohavi, 1995).

7 Evaluation

We evaluated each configuration of Section 5 and each learning method of Section 6 on the dataset which consists of 1336 definitions and 963 non-definitions sentences. Table 7 reports the accuracy and standard error estimated from this experiment.

In all experiment runs, all of the classifiers in all configurations outperform our baseline (75.9%). The best accuracy of each classifier (bold) is between 11.57% to 16.31% above the baseline.

The bigram only attributes (config. 2) clearly outperform the simplest setting (bag-of-word only attributes) for all classifiers. The combination of both attributes (config. 3) achieves some improvement between 0.17% to 4.41% from configuration 2. It is surprising that naive Bayes shows the best and relatively high accuracy in this base configuration (89.82%) and even outperforms all other settings.

Adding syntactic properties (config. 4) or position of sentences in documents (config. 6) to the base configuration clearly gives some improvement (in 4 and 5 classifiers respectively for each configuration). But, adding root forms (config. 7) does not significantly contribute to an improvement. These results show that in general, syntactic properties can improve the performance of most classifiers. The results also support the intuition that the position of sentences in documents plays important role in identifying definition sentences. Moreover, this intuition is also supported by the result that the best performance of naive Bayes is achieved at configuration 6 (90.26%). Compared to the syntactic features, sentence positions give better accuracy in all classifiers.

The above results demonstrate an interesting finding that a simple attribute set which consists of bag-of-words, bigrams, and sentence position under a fast and simple classifier (e.g. naive Bayes) could give a relatively high accuracy. One explanation that we can think of is that candidate sentences have been syntactically very well extracted with our filter. Thus, the sentences are biased by the filter from which important words and bigrams of definitions can be found in most of the sen-

tences. For example, the word and bigrams *is een (is a), een (a), zijn (are), is (is), zijn de (are the),* and *is van (is of)* are good clues to definitions and consequently have high information gain. We have to test this result in a future work on candidate definition sentences which are extracted by filters using various other syntactic patterns.

More improvement is shown when both syntactic properties and sentence position are added together (config. 8). All of the classifiers in this configuration obtain more error reduction compared to the base configuration. Moreover, the best accuracy of this experiment is shown by maximum entropy at this configuration (92.21%). This may be a sign that our proposed syntactic properties are good indicators to identify definition sentences.

Other interesting findings can be found in the addition of named entity classes to configuration 3 (config. 5), to configuration 8 (config. 9) and to configuration 10 (config. 11). In these configurations, adding NEC increases accuracies of almost all classifiers. On the other hand, adding root forms to configuration 3 (config. 7) and to configuration 8 (config. 10) does not improve accuracies. However, the best accuracies of naive Bayes (90.26%) and maximum entropy (92.21%) are achieved when named entity and root forms are not included as attributes.

We now evaluate the classifiers. It is clear from the table that SVM1 and SVM2 settings can not achieve better accuracy compared to the naive Bayes setting, while SVM3 setting marginally outperforms naive Bayes (on 6 out of 11 configurations). This result is contrary to the superiority of SVMs in many text classification tasks. Huang et al. (2003) reported that both classifiers show similar predictive accuracy and AUC (area under the ROC (Receiver Operating Characteristics) curve) scores. This performance of naive Bayes supports the motivation behind its renaisance in machine learning (Lewis, 1998).

From the three SVM settings, SVM with RBF kernel appears as the best classifier for our task in which it outperforms other SVMs settings in all configurations. This result supports the above mentioned argument that if the best C and γ can be selected, we do not need to consider linear SVM (e.g. the svm1 setting).

Among all of the classifiers, maximum entropy shows the best accuracy. It wins at 9 out of 11 configurations in all experiments. This result confirms previous reports e.g. in Nigam et al. (1999) that maximum entropy performs better than naive Bayes in some text classification tasks.

8 Conclusions and future work

We have presented an experiment in identifying definition sentences using syntactic properties and learning-based methods. Our method is concentrated on improving the precision of recognizing definition sentences. The first step is extracting candidate definition sentences from a fully parsed text using syntactic properties of definitions. To distinguish definition from non-definition sentences, we investigated several machine learning methods, namely naive Bayes, maximum entropy, and SVMs. We also experimented with several attribute configurations. In this selection, we combine text properties, document properties, and syntactic properties of the sentences. We have shown that adding syntactic properties, in particular the position of subjects in the sentence, type of determiner of each subject and predicative complement, improves the accuracy of most machine learning techniques, and leads to the most accurate result overall.

Our method has been evaluated on a subset of manually annotated data from Wikipedia. The combination of highly structured text material and a syntactic filter leads to a relatively high initial baseline.

Our results on the performance of SVMs do not confirm the superiority of this learning method for (text) classification tasks. Naive Bayes, which is well known from its simplicity, appears to give reasonably high accuracy. Moreover, it achieves a high accuracy on simple attribute configuration sets (containing no syntactic properties). In general, our method will give the best result if all properties except named entity classes and root forms are used as attributes and maximum entropy is applied as a classifier.

We are currently working on using more syntactic patterns to extract candidate definition sentences. This will increase the number of definition sentences that we can identify from text.

References

I. Androutsopoulos and D. Galanis. 2005. A practically unsupervised learning method to identify single-snippet answers to definition questions on the web. In *Human Language Technology Conference*

and Conference on Empirical Methods in Natural Language Processing (HLT-EMNLP 2005), Vancouver, Canada.

S. Blair-Goldensohn, K. McKeown, and A.H. Schlaikjer. 2004. Answering definitional questions: A hybrid approach. In *New Directions in Question Answering*, pages 47–58.

Gosse Bouma, Jori Mur, Gertjan van Noord, Lonneke van der Plas, and Jörg Tiedemann. 2005. Question answering for Dutch using dependency relations. In *Working Notes for the CLEF 2005 Workshop*, Vienna.

Chih-Chung Chang and Chih-Jen Lin, 2001. *LIBSVM: a library for support vector machines*. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proc. of the 14th COLING*, pages 539–545, Nantes, France.

W. Hildebrandt, B. Katz, and J.J. Lin. 2004. Answering definition questions with multiple knowledge sources. In *HLT-NAACL*, pages 49–56.

Jin Huang, Jingjing Lu, and Charles X. Ling. 2003. Comparing naive bayes, decision trees, and svm with auc and accuracy. In *ICDM '03: Proceedings of the Third IEEE International Conference on Data Mining*, Washington, DC, USA. IEEE Computer Society.

Thorsten Joachims. 1998. Text categorization with support vector machines: learning with many relevant features. In Claire N'edellec and C'eline Rouveirol, editors, *Proceedings of ECML-98, 10th European Conference on Machine Learning*, pages 137–142, Chemnitz, DE. Springer Verlag, Heidelberg, DE.

H. Joho and M. Sanderson. 2000. Retrieving descriptive phrases from large amounts of free text. In *CIKM*, pages 180–186.

S. Sathiya Keerthi and Chih-Jen Lin. 2003. Asymptotic behaviors of support vector machines with gaussian kernel. *Neural Comput.*, 15(7):1667–1689.

J.L. Klavans and S. Muresan. 2000. Definder: Rule-based methods for the extraction of medical terminology and their associated definitions from on-line text. In *American Medical Informatics Assoc 2000.*

Ron Kohavi. 1995. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *IJCAI*, pages 1137–1145.

David D. Lewis. 1998. Naive (Bayes) at forty: The independence assumption in information retrieval. In Claire Nédellec and Céline Rouveirol, editors, *Proceedings of ECML-98, 10th European Conference on Machine Learning*, pages 4–15, Chemnitz, DE. Springer Verlag, Heidelberg, DE.

B. Liu, C.W. Chin, and H.T. Ng. 2003. Mining topic-specific concepts and definitions on the web. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 251–260, New York, NY, USA. ACM Press.

Robert Malouf and Gertjan van Noord. 2004. Wide coverage parsing with stochastic attribute value grammars. In *IJCNLP-04 Workshop Beyond Shallow Analyses - Formalisms and statistical modeling for deep analyses*, Hainan.

A McCallum. 2000. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. `http://www.cs.cmu.edu/~mccallum/bow`.

S. Miliaraki and I. Androutsopoulos. 2004. Learning to identify single-snippet answers to definition questions. In *20th International Conference on Computational Linguistics (COLING 2004)*, pages 1360–1366, Geneva, Switzerland. COLING 2004.

K. Nigam, J. Lafferty, and A. McCallum. 1999. Using maximum entropy for text classification. In *IJCAI-99 Workshop on Machine Learning for Information Filtering*, pages 61–67.

Juan C. Sager and M.C. L'Homme. 1994. A model for definition of concepts. *Terminology*, pages 351–374.

Erik Tjong Kim Sang, Gosse Bouma, and Maarten de Rijke. 2005. Developing offline strategies for answering medical questions. In Diego Mollá and José Luis Vicedo, editors, *AAAI 2005 workshop on Question Answering in Restricted Domains*.

An Ontology-Based Approach to Disambiguation of Semantic Relations

Tine Lassen and **Thomas Vestskov Terney**
Department of Computer Science, Roskilde University, Denmark
tlassen@ruc.dk, tvt@ruc.dk

Abstract

This paper describes experiments in using machine learning for relation disambiguation. There have been succesfuld experiments in combining machine learning and ontologies, or light-weight ontologies such as WordNet, for word sense disambiguation. However, what we are trying to do, is to disambiguate complex concepts consisting of two simpler concepts and the relation that holds between them. The motivation behind the approach is to expand existing methods for content based information retrieval. The experiments have been performed using an annotated extract of a corpus, consisting of prepositions surrounded by noun phrases, where the prepositions denote the relation we are trying disambiguate. The results show an unexploited opportunity of including prepositions and the relations they denote, e.g. in content based information retrieval.

1 Introduction

What we describe in this paper, which we refer to as relation disambiguation, is in some sense similar to word sense disambiguation. In traditional word sense disambiguation the objective is to associate a distinguishable sense with a given word (Ide and Véronis, 1998). It is not a novel idea to use machine learning in connection with traditional word sense disambiguation, and as such it is not a novel idea to include some kind of generalization of the concept that a word expresses in the learning task either (Yarowsky, 1992). Other projects have used light-weight ontologies such as WordNet in this kind of learning task (Voorhees, 1993; Agirre and Martinez, 2001). What we believe is our contribution with this work is the fact that we attempt to learn complex concepts that consist of two simpler concepts, and the relation that holds between them. Thus, we start out with the knowledge that some relation holds between two concepts, which we could express as REL(concept1,concept2), and what we aim at being able to do is to fill in a more specific relation type than the generic REL, and get e.g. POF(concept1,concept2) in the case where a preposition expresses a partitive relation. This makes it e.g. possible to determine from the sentence "France is in Europe" that France is a part of Europe. As in word sense disambiguation we here presuppose a finite and minimal set of relations, which is described in greater detail in section 2.

The ability to identify these complex structures in text, can facilitate a more content based information retrieval as opposed to more traditional search engines, where the information retrieval relies more or less exclusively on keyword recognition. In the OntoQuery project[1], pertinent text segments are retrieved based on the conceptual content of the search phrase as well as the text segments (Andreasen et al., 2002; Andreasen et al., 2004). Concepts are here identified through their corresponding surface form (noun phrases), and mapped into the ontology. As a result, we come from a flat structure in a text to a graph structure, which describes the concepts that are referred to in a given text segment, in relation to each other.

However, at the moment the ontology is strictly a subsumption-based hierarchy and, further, only relatively simple noun phrases are recognized and mapped into the ontology. The work presented here expands this scope by including other semantic relations between noun phrases. Our first experiments in this direction have been an analysis of prepositions with surrounding noun phrases (NPs). Our aim is to show that there is an affinity between the ontological types of the NP-heads and the relation that the preposition denotes, which can be used to represent the text as a complex semantic structure, as opposed to simply running text. The approach to showing this has been to annotate a corpus and use standard machine learning methods on this corpus.

2 Semantic relations

The following account is based on the work of (Jensen and Nilsson, 2006): Relations exist between entities referred to in discourse. They can exist at different syntactic levels; across sentence boundaries as in example 1, or within a sentence, a phrase or a word. The relations

[1] http://www.ontoquery.dk

can be denoted by different parts of speech, such as a verb, a preposition or an adjective, or they can be implicitly present in compounds and genitive constructions as in example 2.

Semantic relations are n-ary: In example 1 below the verb form 'owns' denotes a binary relation between *Peter* and *a dog*, and in example 3, the verb form 'gave' denotes a ternary relation between *Peter*, *the dog* and *a bone*. In example 4 the preposition 'in' denotes a binary relation between *the dog* and *the yard*.

(1) Peter owns a dog. It is a German shepherd.

(2) Peter's dog.

(3) Peter gave the dog a bone.

(4) The dog in the yard.

In the framework of this machine learning project, we will only consider binary relations denoted by prepositions. A preposition, however, can be ambiguous in regard to which relation it denotes. As an example, let us consider the Danish preposition *i* (Eng: in): The surface form *i* in 'A i B' can denote at least five different relations between A and B:

1. A patient relation *PNT*; a relation where one of the arguments' case role is patient, e.g. *"ændringer i stofskiftet"* (changes in the metabolism).
2. A locational relation *LOC*; a relation that denotes the location/position of one of the arguments compared to the other argument, e.g. *"skader i hjertemuskulaturen"* (injuries in the heart muscle).
3. A temporal relation *TMP*; a relation that denotes the placement in time of one of the arguments compared to the other, e.g. *"mikrobiologien i 1800-tallet"* (microbiology in the 19th century).
4. A property ascription relation *CHR*; a relation that denotes a characterization relation between one of the arguments and a property, e.g. *"antioxidanter i renfremstillet form"* (antioxidants in a pure form)
5. A 'with respect to' relation *WRT*; an underspecified relation that denotes an 'aboutness' relation between the arguments, e.g. *"forskelle i saltindtagelsen"* (differences in the salt intake) .

As presented above, the idea is to perform supervised machine learning, that will take into account the surface form of the preposition and the ontological type of the heads of the surrounding noun phrases, and on this basis be able to determine the relation that holds between noun phrases surrounding a preposition in unseen text.

3 The corpus

In order to establish a training set, a small corpus of approximately 18,500 running words has been compiled from texts from the domain of nutrition and afterwards annotated with the ontological type of the head of the noun phrases, and the semantic relation denoted by the preposition [2].

All the text samples in this corpus derive from "The Danish National Encyclopedia" (Gyldendal, 2004), and are thus not only limited domain-wise, but also of a very specific text type which can be classified as expert-to-non-expert. Thus, we cannot be certain that our results can be directly transferred to a larger or more general domain, or to a different text type. This aspect would have to be empirically determined.

3.1 Annotation

For the purpose of learning relations, 952 excerpts of the form:

$$NP - P - NP \tag{5}$$

have been extracted from the corpus and annotated with information about part of speech, ontological type and relation type for NP heads and prepositions, respectively. An example of the analyzed text excerpts are given in table 1 on the following page, where each row indicates a level of the analysis.

The POS-tagging and head extraction have been done automatically, the ontological type assignation partly automatically (ontology look-up) and partly manually (for words that do not exist as instantiations of concepts in the ontology). The relation annotation has been done manually.

The tags used in the annotation on the three levels are:

POS-tags. Our tagger uses a subset of the PAROLE tag set, consisting of 43 tags, see (Hansen, 2000), which means that it is a low level POS tagging with little morphosyntactic information. We only use the tags in order to extract NPs and prepositions, and thus do not need a more fine-grained information level.

SIMPLE-tags. The tags used for the ontological type annotation consist of abbreviations of the types in the SIMPLE top ontology. The tag set consists of 151 tags.

Relation-tags. The tags used for the relation annotation derive from a minimal set of relations that have been used in earlier OntoQuery related work. The set can be seen in table 2

[2] Extraction, POS-tagging and initial ontological and relation type annotation was done by Dorte Haltrup Hansen, CST, University of Copenhagen

surface form	*blodprop* (thrombosis)	*i* (in)	*hjertet* (the heart)
syntactic structure	head of first NP	preposition	head of second NP
relation and ontological type	disease	location	body part

Table 1: Example of the text excerpts analyzed in our experiments. Each row indicate a level of analysis

The manual relation annotation has been done by one annotator for this initial project. The ideal situation would be to have several annotators annotate the corpus. If two or more people annotate the same corpus, they are almost certain to disagree on some occasions. This disagreement can have two sources: first it can be due to cognitive differences. Two people subjected to the same utterance are not guaranteed to perceive the same content, or to perceive the content intended by the producer of the utterance. Many factors are at play here; cultural background, knowledge, memory, etc.

Secondly, it can be due to conceptual, lexical or syntactic ambiguity in the utterance. We cannot remove these sources of disagreement, but we can introduce tools that make the annotation more consistent. By using a finite and minimal realtion tag set and, further, by introducing paraphrase tests, we hope to minimize the risk of inter-annotator disagreement in a future annotation on a larger scale.

3.1.1 The ontological type annotation

As noted above, the ontological types used in the experiments derive from the SIMPLE top ontology (Pedersen, 1999; Lenci et al., 2000). The heads of the phrases have been annotated with the lowest possible node, i.e. ontological type, of the top ontology. In the case of *blodprop* the annotation of ontological type is "disease", since "disease" is the lowest node in the top ontology in the path from thrombosis to the top. This is illustrated in figure 1, which shows the path from *blodprop* (thrombosis) to the top level of SIMPLE.

Thus, for the purpose of this project, we only consider one node for each concept: the lowest possible node in the top ontology. Another approach would be to consider the the full path to the top node, and also including the path from the leaf node to the lowest node in the top ontology. In the example depicted in figure 1, the full path from trombosis to the top node would be *trombosis–cardiovascular disease–disease–phenomenon–event–entity–top* or *trombosis–cardiovascular disease–disease–agentive–top*.

3.1.2 The set of relations

For the purpose of the manual relation annotation, we needed to decide on a finite set of possible relations that can be denoted by prepositions. This is a non-trivial task, as it is almost impossible to foresee which relations prepositions *can* denote generally, and in the text type at hand specifically, by introspection alone. The method that we decided to use was the following: An initial set of relations that have all been used in prior OntoQuery-related work (Nilsson, 2001; Madsen et al., 2001; Madsen et al., 2000), were chosen as a point of departure. The final set was found by annotating the text segments using this set as the possible relation types, and the relations that are actually manifested in the data then form the final subset that was used as input for a machine learning algorithm. The final subset is shown in table 2.

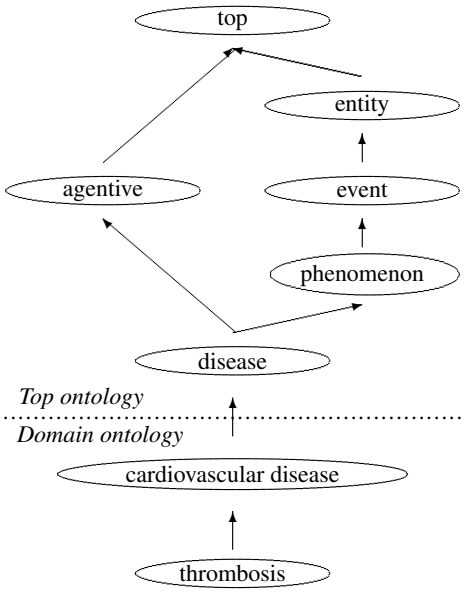


Figure 1: An illustration of the path from *blodprop* (thrombosis) to the top level of the SIMPLE ontology.

Role	Description
AGT	Agent of act or process
BMO	By means of, instrument, via
CBY	Caused by
CHR	Characteristic (property ascription)
CMP	Comprising, has part
DST	Destination of moving process
LOC	Location, position
PNT	Patient of act or process
SRC	Source of act or process
TMP	Temporal aspects
WRT	With respect to

Table 2: The set of relations used in the annotation, which is a subset of the set proposed in Nilsson, 2001.

3.2 Paraphrase tests

In order to ensure a consistent relation annotation, it is necessary to develop a set of paraphrase tests that can help the annotator determine which relation a given preposition denotes in a given context. Some relations are particularly difficult to intuitively keep apart from closely related relations. One of these problematic relation pairs is treated in some detail below.

For example locative and partitive relations can be difficult to keep apart, probably because they to some extent are overlapping semantically. From a philosophical point of view, an important question is 'when does an entity become part of the entity it is located in?', but from a practical point of view, we are interested in answering the question 'how can we decide if a given relation a locative or partitive relation?'.

In this paper we will only treat the latter question. A tool that is useful for this purpose is the paraphrase test: If we can paraphrase the text segment in question into the phrasing the test prescribes, while preserving the semantic content, we can conclude that the relation is a possible relation for the given phrase.

3.2.1 Attribute Transportation Test

The two relations LOC and POF can be difficult to differentiate, even when using paraphrase tests. Therefore, an additional test that could be considered, is Ruus' attribute transportation test (Ruus, 1995)[3]. In the example "The pages in the book", the book gets e.g. the attribute 'binding: {hardback | paperback}' from *cover*, and the attribute 'paper grade:{bond | book | bristol | newsprint}' from *pages*.

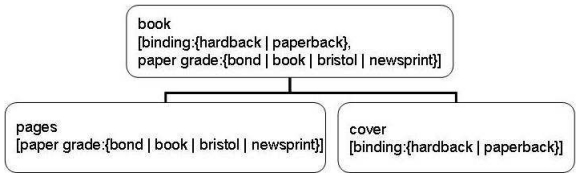


Figure 2: A graphical representation of the relation between book and pages

We cannot observe an attribute transport, neither from the bird to the roof, nor the other way. This suggests that it is possible to use the atrribute transportation test in order to determine whether a given relation is a POF or a LOC relation. Thus, we can now formulate the following paraphrase test for POF:

POF: *A consists e.g. of B* **and**
A has the attribute X, from B.

[3]We will here ignore the question of direction of transport

4 Experiments

The annotation process generates af a feature space of six dimensions, namely the lemmatized form of the two heads of the noun phrases, the ontological types of the heads, the preposition and the relation. In the corpus there is a total of only 952 text segments. In general the distribution of the data is highly skewed and sparseness is a serious problem. More than half of the instances are of the relation type WRT or PNT, and the rest of the instances are distributed among the remaining 10 relations with only 14 instances scattered over the tree smallest classes. This is illustrated in figure 3. There are 332 different combinations of ontological types where 197 are unique. There are 681 different heads and 403 of them are unique, with all of them being lemmatized.

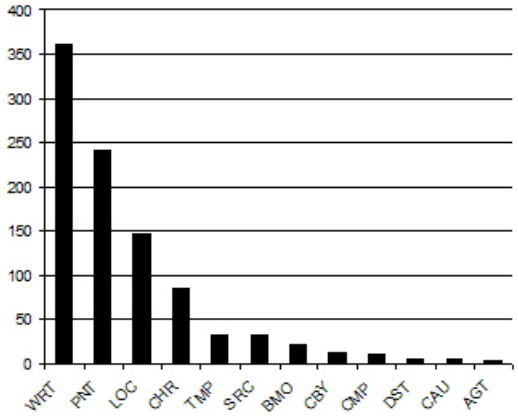


Figure 3: An illustration of the distribution of the 12 possible relations.

Our assumption is that there is consistency in which relations prepositions usually denote in particular contexts, and hence the learning algorithms should be able to generalize well. We also assume that the addition of the ontological types of the head of the NP, is the most vital information in classifying the relation type, at least in this case where data is sparse.

We have run the experiments with a Support Vector Machine algorithm SMO (Keerthi et al., 2001) and the prepositional rule learning algorithm JRip (Cohen, 1995). The former in order to get high precision, the latter in order to get easily interpretable rules for later analysis (see section 4.1). The experiments were run using 10-fold-cross-validation, with a further partition of the training set at each fold into a tuning and a training set. The tuning set was used to optimize the parameter[4] settings for each algorithm . The implementation of the algorithms that we used, was the WEKA software package (Frank et al., 2005).

[4]For SMO the parameters where complexity, kernel used and gamma for the RBF kernel. For JRip it was number of folds used for growing and pruning, minimum number of instances covered and number of optimization runs

The experiments were run on seven different combinations of the feature space, ranging from using only the heads to using both heads, preposition and ontological types of the heads. This was done in order to get insight into the importance of using ontological types in the learning. The results of these experiments are shown in table 3. The last column shows the precision for a projected classifier (PC) in the cases where it outperforms the trivial rejector. The projected classifier, in this case, assigns the relation that is most common for the corresponding input pair; e.g if the ontological types are DIS/HUM, then the most common relation is PNT. The trivial rejector, which assigns the most common relation, in this case WRT, to all the instances, achieves a precision of 37.8%.

Feature space		JRip	SVM	PC
1	Preposition	68.4	68.5	67.6
2	Ontological types	74.4	77.0	61.8
3	Lemma	66.8	73.3	–
4	Lemma and Preposition	72.3	83.4	–
5	Ontological types and Lemma	74,7	81.7	–
6	Ontological types and Preposition	82.6	86.6	–
7	Ontological types, Preposition and Lemma	84,0	88.3	–

Table 3: The precision of SVM, JRip and a projected classifier on the seven different combinations of input features. "Lemma" here is short for lemmatized NP head.

The following conclusions can be drawn from table 3. The support vector machine algorithm produces a result which in all cases is better than the baseline, i.e. we are able to produce a model that generalizes well over the training instances compared to the projected classifier or the trivial rejector. This difference is not statistically significant at a confidence level of 0.95 when only training on the surface form of prepositions.

A comparison of line 1–3 shows that training on ontological types seems to be superior to using lemmatized NP heads or prepositions, though the superiority is not statistically significant when comparing to the lemmatized NP heads. When comparing line 4–7 the difference between the results are not statistically significant. This fact may owe to the data sparseness. However, comparing line 1 to line 6 or 7, shows that the improvement of adding the preposition and the lemmatized NP heads to the ontological types is statistically significant.

In general, the results reveal an unexplored opportunity to include ontological types and the relations that prepositions denote in information retrieval. In the next section, we will look more into the rules created by the JRip algorithm from a linguistic point of view.

4.1 Analyzing the rules

In this section we will take a deeper look into the rules produced by JRip on the data set with only ontological types, since they are the most interesting in this context.

The JRip algorithm produced on average 21 rules. The most general rule covering almost half of the instances is the default rule, that assigns all instances to the WRT relation if no other rules apply. At the other end of the spectrum, there are ten rules covering no more than 34 instances, but with a precision of 100%. It is futile to analyse these rules, since they cover the most infrequent relations and hence may be overfitting the data set. However, this seems not be the case with a rule like "*if* the ontotype of the first head is DISEASE and and the ontotype of the second head is HUMAN *then* the relation is PATIENT" covering an instance as e.g. "iron deficiency in females".

The rule with the second highest coverage, and a fairly low precision of around 66%, is the rule: "*if* the ontotype of the second head is BODY PART *then* the relation type is LOCATIVE". The rule covers instances as e.g. "...thrombosis in the heart" but also incorrectly classifies all instances as LOCATIVE where the relation type should be SOURCE. E.g. the sentence '...iron absorbtion from the intestine", which is in fact a SOURCE relation, but is classified as LOCATIVE by the rule.

One of the least surprising and most precise rules is: "*if* the ontotype of the second head is TIME *then* the relation type is TEMPORAL" covering an instance as e.g. "...diet for many months". We would expect a similar rule to be produced, if we had performed the learning task on a general language corpus.

5 Conclusion and future work

Even though the experiments are in an early phase, the results indicate that it is possible to analyse the semantic relation a preposition denotes between two noun phrases, by using machine learning and an annotated corpus – at least within the domain covered by the ontology. Future work will therefore include annotation and investigation of a general language corpus. Also, a more thorough examination of the corpus, more specifically an investigation of which relations or prepositions that are most difficult to analyse. Also, we will experiment with the amount of information that we train on, not as we have already done by in- or excluding *types* of information, but rather the extension of the information: Could we predict the ontological type of one of the arguments by looking at the other? Finally, an explicit inclusion of the whole ontology in the learning process is on the agenda, as proposed in section 3.1.1 on page 3, in the anticipation that the learner will produce an even better model.

6 Acknowledgements

We would like to thank Troels Andreasen, Per Anker Jensen and two anonymous reviewers for fruitful comments. The latter especially for comments on the experimental part and inter-annotator agreement.

References

[Agirre and Martinez2001] E. Agirre and D. Martinez. 2001. Learning class-to-class selectional preferences.

[Andreasen et al.2002] Troels Andreasen, Per Anker Jensen, Jørgen Fischer Nilsson, Patrizia Paggio, Bolette Sandford Pedersen, and Hanne Erdman Thomsen. 2002. Ontological extraction of content for text querying. In *Lecture Notes in Computer Science*, volume 2553, pages 123 – 136. Springer-Verlag.

[Andreasen et al.2004] Troels Andreasen, Per Anker Jensen, Jørgen Fischer Nilsson, Patrizia Paggio, Bolette Sandford Pedersen, and Hanne Erdman Thomsen. 2004. Content-based text querying with ontological descriptors. *Data & Knowledge Engineering*, 48(2):199–219.

[Cohen1995] William W. Cohen. 1995. Fast effective rule induction. In Armand Prieditis and Stuart Russell, editors, *Proceedings of the 12th International Conference on Machine Learning*, pages 115–123, Tahoe City, CA. Morgan Kaufmann.

[Frank et al.2005] Eibe Frank, Mark Hall, and Len Trigg. 2005. Weka. Publicly available, November.

[Gyldendal2004] Gyldendal. 2004. The danish national encyclopedia. ISBN: 8702031051.

[Hansen2000] Dorte Haltrup Hansen. 2000. Træning og brug af brill-taggeren på danske tekster. Technical report, CST.

[Ide and Véronis1998] Nancy Ide and Jean Véronis. 1998. Special issue on word sense disambiguation: Introduction to the special issue on word sense disambiguation: the state of the art. *Computational Linguistics*, 24.

[Jensen and Nilsson2006] Per Anker Jensen and Jørgen Fischer Nilsson, 2006. *Syntax and Semantics of Prepositions*, volume 29 of *Text, Speech and Language Technology*, chapter Ontology-Based Semantics for Prepositions. Springer.

[Keerthi et al.2001] S. Sathiya Keerthi, Shirish Krishnaj Shevade, Chiranjib Bhattacharyya, and K. R. K. Murthy. 2001. Improvements to platt's smo algorithm for svm classifier design. *Neural Computation*, 13(3):637–649.

[Lenci et al.2000] Alessandro Lenci, Nuria Bel, Federica Busa, Nicoletta Calzolari1, Elisabetta Gola, Monica Monachini, Antoine Ogonowski, Ivonne Peters, Wim Peters, Nilda Ruimy, Marta Villegas, and Antonio Zampolli. 2000. Simple: A general framework for the development of multilingual lexicons. *International Journal of Lexicography*, 13(4):249–263.

[Madsen et al.2000] Bodil Nistrup Madsen, Bolette Sandford Pedersen, and Hanne Erdman Thomsen. 2000. Semantic relations in content-based querying systems: a research presentation from the ontoquery project. In K Simov and A Kiryakov, editors, *Ontologyes and Lexical Knowledge Bases. Proceedings of the 1st International Workshop, OntoLex 2000*. University of Southern Denmark, Kolding.

[Madsen et al.2001] Bodil Nistrup Madsen, Bolette Sandford Pedersen, and Hanne Erdman Thomsen. 2001. Defining semantic relations for ontoquery. In Per Anker Jensen and P Skadhauge, editors, *Proceedings of the First International OntoQuery Workshop Ontology-based interpretation of NP's*. University of Southern Denmark, Kolding.

[Nilsson2001] Jørgen Fischer Nilsson. 2001. A logico-algebraic framework for ontologies, ontolog. In Jensen and Skadhauge, editors, *Proceedings of the First International OntoQuery Workshop Ontology-based interpretation of NP's*. University of Southern Denmark, Kolding.

[Pedersen1999] Bolette Sandford Pedersen. 1999. Den danske simple-ordbog. en semantisk, ontologibaseret ordbog. In C. Poulsen, editor, *DALF 99, Datalingvistisk Forenings årsmøde 1999*. Center for sprogteknologi.

[Ruus1995] Hanne Ruus. 1995. *Danske kerneord. Centrale dele af den danske leksikalske norm 1-2*. Museum Tusculanums Forlag.

[Voorhees1993] Ellen M. Voorhees. 1993. Using wordnet to disambiguate word senses for text retrieval. In Robert Korfhage, Edie M. Rasmussen, and Peter Willett, editors, *Proceedings of the 16th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval. Pittsburgh, PA, USA, June 27 - July 1, 1993*, pages 171–180. ACM.

[Yarowsky1992] David Yarowsky. 1992. Word-sense disambiguation using statistical models of Roget's categories trained on large corpora. In *Proceedings of COLING-92*, pages 454–460, Nantes, France, July.

Towards Free-text Semantic Parsing: A Unified Framework Based on FrameNet, VerbNet and PropBank

Ana-Maria Giuglea and **Alessandro Moschitti**
University of Rome "Tor Vergata",
Rome, Italy
ana-maria.giuglea@topex.ro
moschitti@info.uniroma2.it

Abstract

This article describes a robust semantic parser that uses a broad knowledge base created by interconnecting three major resources: FrameNet, VerbNet and PropBank. The FrameNet corpus contains the examples annotated with semantic roles whereas the VerbNet lexicon provides the knowledge about the syntactic behavior of the verbs. We connect VerbNet and FrameNet by mapping the FrameNet frames to the VerbNet Intersective Levin classes. The PropBank corpus, which is tightly connected to the VerbNet lexicon, is used to increase the verb coverage and also to test the effectiveness of our approach. The results indicate that our model is an interesting step towards the design of free-text semantic parsers.

1 Introduction

During the last years a noticeable effort has been devoted to the design of lexical resources that can provide the training ground for automatic semantic role labelers. Unfortunately, most of the systems developed until now are confined to the scope of the resource that they use during the learning stage. A very recent example in this sense was provided by the CONLL 2005 Shared Task on PropBank (Kingsbury and Palmer, 2002) role labeling (Carreras and Màrquez, 2005). While the best F-measure recorded on a test set selected from the training corpus (WSJ) was 80%, on the Brown corpus, the F-measure dropped below 70%. The most significant causes for this performance decay were highly ambiguous and unseen predicates (i.e. predicates that do not have training examples, unseen in the training set).

On the FrameNet (Johnson et al., 2003) role labeling task, the Senseval-3 competition (Litkowski, 2004) registered similar results (~80%) by using the gold frame information as a given feature. No tests were performed outside FrameNet. In this paper, we show that when the frame feature is not used, the performance decay on different corpora reaches 30 points. Thus, the context knowledge provided by the frame is very important and a free-text semantic parser using FrameNet roles depends on the accurate automatic detection of this information.

In order to test the feasibility of such a task, we have trained an SVM (Support Vector Machine) Tree Kernel model for the automatic acquisition of the frame information. Although FrameNet contains three types of predicates (nouns, adjectives and verbs), we concentrated on the verb predicates and the roles associated with them. Therefore, we considered only the frames that have at least one verb lexical unit. Our experiments show that given a FrameNet predicate-argument structure, the task of identifying the originating frame can be performed with very good results when the verb predicates have enough training examples, but becomes very challenging otherwise. The predicates not yet included in FrameNet and the predicates belonging to new application domains (that require new frames) are especially problematic as for them there is no available training data.

We have thus studied new means of capturing the semantic context, other than the frame, which can be easily annotated on FrameNet and are available on a larger scale (i.e. have a better coverage). A very good candidate seems to be the Intersective Levin classes (Dang et al., 1998) that can be found as well in other predicate resources like PropBank and VerbNet (Kipper et al., 2000). Thus, we have designed a semi-automatic algorithm for assigning an Intersective Levin class to each FrameNet verb predicate.

The algorithm creates a mapping between FrameNet frames and the Intersective Levin classes. By doing that we could connect FrameNet to VerbNet and PropBank and obtain an *increased training set* for the Intersective Levin class. This leads to better verb coverage and a more robust semantic parser. The newly created knowledge base allows us to surpass the shortcomings that arise when FrameNet, VerbNet and PropBank are used separately while, at the same time, we benefit from the extensive research involving each of them (Pradhan et al., 2004; Gildea and Jurafsky, 2002; Moschitti, 2004).

We mention that there are 3,672 distinct verb senses[1] in PropBank and 2,351 distinct verb senses in FrameNet. Only 501 verb senses are in common between the two corpora which mean 13.64% of PropBank and 21.31% of FrameNet. Thus, by training an Intersective Levin class classifier on both PropBank and FrameNet we extend the number of available verb senses to 5,522.

In the remainder of this paper, Section 2 summarizes previous work done on FrameNet automatic role detection. It also explains in more detail why models based exclusively on this corpus are not suitable for free-text parsing. Section 3 focuses on VerbNet and PropBank and how they can enhance the robustness of our semantic parser. Section 4 describes the mapping between frames and Intersective Levin classes whereas Section 5 presents the experiments that support our thesis. Finally, Section 6 summarizes the conclusions.

2 Automatic semantic role detection on FrameNet

One of the goals of the FrameNet project is to design a linguistic ontology that can be used for automatic processing of semantic information. This hierarchy contains an extensive semantic analysis of verbs, nouns, adjectives and situations in which they are used, called *frames*. The basic assumption on which the frames are built is that each word evokes a particular situation with specific participants (Fillmore, 1968). The situations can be fairly simple depicting the entities involved and the roles they play or can be very complex and in this case they are called *scenarios*. The word that evokes a particular frame is called *target word* or *predicate* and can be an adjective, noun or verb. The participant entities are defined using semantic roles and they are called *frame elements*.

Several models have been developed for the automatic detection of the frame elements based on the FrameNet corpus (Gildea and Jurafsky, 2002; Thompson et al., 2003; Litkowski, 2004). While the algorithms used vary, almost all the previous studies divide the task into 1) the identification of the verb arguments to be labeled and 2) the tagging of each argument with a role. Also, most of the models agree on the core features as being: *Predicate*, *Headword*, *Phrase Type*, *Governing Category*, *Position*, *Voice* and *Path*. These are the initial features adopted by Gildea and Jurafsky (2002) (henceforth G&J) for both frame element identification and role classification.

A difference among the previous machine-learning models is whether the frame information was used as gold feature. Of particular interest for us is the impact of the frame over unseen predicates and unseen words in general. The results obtained by G&J are relevant in this sense; especially, the experiment that uses the frame to generalize from predicates seen in the training data to other predicates (i.e. when no data is available for a target word, G&J use data from the corresponding frame). The overall performance induced by the frame usage increased.

Other studies suggest that the frame is crucial when trying to eliminate the major sources of errors. In their error analysis, (Thompson et al., 2003) pinpoints that the verb arguments with headwords that are "rare" in a particular frame but not rare over the whole corpus are especially hard to classify. For these cases the frame is very important because it provides the context information needed to distinguish between different word senses.

Overall, the experiments presented in G&J's study correlated with the results obtained in the Senseval-3 competition show that the frame feature increases the performance and decreases the amount of annotated examples needed in training (i.e. frame usage improves the generalization ability of the learning algorithm). On the other hand the results obtained without the frame information are very poor.

This behavior suggests that predicates in the same frame behave similarly in terms of their argument structure and that they differ with respect to other frames. From this perspective, having a broader verb knowledge base becomes of major importance for free-text semantic parsing.

[1] A verb sense is an Intersective Levin class in which the verb is listed.

Unfortunately, the 321 frames that contain at least one verb predicate cover only a small fraction of the English verb lexicon and of possible domains. Also from these 321 frames only 100 were considered to have enough training data and were used in Senseval-3 (see Litkowski, 2004 for more details).

Our approach for solving such problems involves the usage of a frame-like feature, namely the Intersective Levin class. We show that the Levin class is similar in many aspects to the frame and can replace it with almost no loss in performance. At the same time, Levin class provides better coverage as it can be learned also from other corpora (i.e. PropBank). We annotate FrameNet with Intersective Levin classes by using a mapping algorithm that exploits current theories of linking. Our extensive experimentation shows the validity of our technique and its effectiveness on corpora different from FrameNet. The next section provides the theoretical support for the unified usage of FrameNet, VerbNet and PropBank, explaining why and how is possible to link them.

3 Linking FrameNet to VerbNet and PropBank

In general, predicates belonging to the same FrameNet frame have a coherent syntactic behavior that is also different from predicates pertaining to other frames (G&J). This finding is consistent with theories of linking that claim that the syntactic behavior of a verb can be predicted from its semantics (Levin 1993, Levin and Rappaport Hovav, 1996). This insight determined us to study the impact of using a feature based on Intersective Levin classes instead of the frame feature when classifying FrameNet semantic roles. The main advantage of using Levin classes comes from the fact that other resources like PropBank and the VerbNet lexicon contain this kind of information. Thus, we can train a Levin class classifier also on the PropBank corpus, considerably increasing the verb knowledge base at our disposal. Another advantage derives from the syntactic criteria that were applied in defining the Levin clusters. As shown later in this article, the syntactic nature of these classes makes them easier to classify than frames, when using only syntactic and lexical features.

More precisely, the Levin clusters are formed according to diathesis alternation criteria which are variations in the way verbal arguments are grammatically expressed when a specific semantic phenomenon arises. For example, two different types of diathesis alternations are the following:

(a) **Middle Alternation**
[$_{\text{Subject, Agent}}$ The butcher] cuts [$_{\text{Direct Object, Patient}}$ the meat].
[$_{\text{Subject, Patient}}$ The meat] cuts easily.

(b) **Causative/inchoative Alternation**
[$_{\text{Subject, Agent}}$ Janet] broke [$_{\text{Direct Object, Patient}}$ the cup].
[$_{\text{Subject, Patient}}$ The cup] broke.

In both cases, what is alternating is the grammatical function that the Patient role takes when changing from the transitive use of the verb to the intransitive one. The semantic phenomenon accompanying these types of alternations is the change of focus from the entity performing the action to the theme of the event.

Levin documented 79 alternations which constitute the building blocks for the verb classes. Although alternations are chosen as the primary means for identifying the classes, additional properties related to subcategorization, morphology and extended meanings of verbs are taken into account as well. Thus, from a syntactic point of view, the verbs in one Levin class have a regular behavior, different from the verbs pertaining to other classes. Also, the classes are semantically coherent and all verbs belonging to one class share the same participant roles.

This constraint of having the same semantic roles is further ensured inside the VerbNet lexicon that is constructed based on a more refined version of the Levin classification called Intersective Levin classes (Dang et al., 1998). The lexicon provides a regular association between the syntactic and semantic properties of each of the described classes. It also provides information about the syntactic frames (alternations) in which the verbs participate and the set of possible semantic roles.

One corpus associated with the VerbNet lexicon is PropBank. The annotation scheme of PropBank ensures that the verbs belonging to the same Levin class share similarly labeled arguments. Inside one Intersective Levin class, to one argument corresponds one semantic role numbered sequentially from Arg0 to Arg5. Higher numbered argument labels are less consistent and assigned per-verb basis.

The Levin classes were constructed based on regularities exhibited at grammatical level and the resulting clusters were shown to be semantically coherent. As opposed, the FrameNet frames were build on semantic bases, by putting together verbs, nouns and adjectives that evoke the same situations. Although different in conception, the

FrameNet verb clusters and VerbNet verb clusters have common properties[2]:

(1) Coherent syntactic behavior of verbs inside one cluster,
(2) Different syntactic properties between any two distinct verb clusters,
(3) Shared set of possible semantic roles for all verbs pertaining to the same cluster.

Having these insights, we have assigned a correspondent VerbNet class not to each verb predicate but rather to each frame. In doing this we have applied the simplifying assumption that a frame has a unique corresponding Levin class. Thus, we have created a one-to-many mapping between the Intersective Levin classes and the frames. In order to create a pair ⟨FrameNet frame, VerbNet class⟩, our mapping algorithm checks both the syntactic and semantic consistency by comparing the role frequency distributions on different syntactic positions for the two candidates. The algorithm is described in detail in the next section.

4 Mapping FrameNet frames to VerbNet classes

The mapping algorithm consists of three steps: (a) we link the frames and Intersective Levin verb classes that have the largest number of verbs in common and we create a set of pairs ⟨FrameNet frame, VerbNet class⟩ (see Figure 1); (b) we refine the pairs obtained in the previous step based on diathesis alternation criteria, i.e. the verbs pertaining to the FrameNet frame have to undergo the same diathesis alternation that characterize the corresponding VerbNet class (see Figure 2) and (c) we manually check and correct the resulting mapping. In the next sections we will explain in more detail each step of the mapping algorithm.

4.1 Linking frames and Intersective Levin classes based on common verbs

During the first phase of the algorithm, given a frame, we compute its intersection with each VerbNet class. We choose as candidate for the mapping the Intersective Levin class that has the largest number of verbs in common with the given frame (Figure 1, line (I)). If the size of the intersection between the FrameNet frame and the candidate VerbNet class is bigger than or equal to 3 elements then we form a pair ⟨FrameNet frame, VerbNet class⟩ that qualifies for the second step of the algorithm.

Only the frames that have more than three verb lexical units are candidates for this step (frames with less than 3 members cannot pass condition (II)). This excludes a number of 60 frames that will subsequently be mapped manually.

INPUT
$VN = \{C \mid C \text{ is a VerbNet class}\}$
$VN\ Class\ C = \{v \mid v \text{ is a verb of } C\}$
$FN = \{F \mid F \text{ is a FrameNet frame}\}$
$FN\ Frame\ F = \{v \mid v \text{ is a verb of } F\}$
OUTPUT
$Pairs = \{(F,C) \mid F \in FN, C \in VN : F \text{ is mapped to } C\}$
COMPUTE PAIRS :
Let $Pairs = \varnothing$
for each $F \in FN$
(*I*) *compute* $C^* = \arg\max_{C \in VN} |F \cap C|$
(*II*) *if* $|F \cap C^*| \geq 3$ *then* $Pairs = Pairs \cup (F, C^*)$

Figure 1. Linking FrameNet frames and VerbNet classes

4.2 Refining the mapping based on verb alternations

In order to assign a VerbNet class to a frame, we have to check that the verbs belonging to that frame respect the diathesis alternation criteria used to define the VerbNet class. Thus, the pairs ⟨FrameNet frame, VerbNet class⟩ formed in step (I) of the mapping algorithm have to undergo a validation step that verifies the similarity between the enclosed FrameNet frame and VerbNet class. This validation process has several substeps.

First, we make use of the property (3) of the Levin classes and FrameNet frames presented in the previous section. According to this property, all verbs pertaining to one frame or Levin class have the same participant roles. Thus, a first test of compatibility between a frame and a Levin class is that they share the same participant roles. As FrameNet is annotated with frame-specific semantic roles we manually mapped these roles into the VerbNet set of thematic roles. Given a frame, we assigned thematic roles to all frame elements that are associated with verbal predicates. For example the roles *Speaker*, *Addressee*, *Message* and *Topic* from the *Telling* frame were respectively mapped into *Agent*, *Recipient*, *Theme* and *Topic*.

[2] For FrameNet, properties 1 and 2 are true for most of the frames but not for all. See section 4.4 for more details.

Second, we build a frequency distribution of VerbNet thematic roles on different syntactic position. Based on our observation and previous studies (Merlo and Stevenson, 2001), we assume that each Levin class has a distinct frequency distribution of roles on different grammatical slots. As we do not have matching grammatical function in FrameNet and VerbNet, we approximate that *subjects* and *direct objects* are more likely to appear on positions adjacent to the predicate, while *indirect objects* appear on more distant positions. The same intuition is used successfully by G&J in the design of the *Position* feature.

We will acquire from the corpus, for each thematic role θ_i, the frequencies with which it appears on an adjacent (ADJ) or distant (DST) position in a given frame or VerbNet class (i.e. #(θ_i, class, position)). Therefore, for each frame and class, we obtain two vectors with thematic role frequencies corresponding respectively to the adjacent and distant positions (see Figure 2). We compute a score for each pair ⟨FrameNet frame, VerbNet class⟩ using the normalized scalar product. We give a bigger weight to the adjacent dot product multiplying its score by 2/3 with respect to the distant dot product that is multiplied by 1/3. We do this to minimize the impact that adjunct roles like Temporal and Location (that appear mostly on the distant positions) could have on the final outcome.

$$TR=\{\theta_i:\ \theta_i \text{ is the } i^{th} \text{ theta role of the VerbNet theta role set}\}$$
$$\text{for each } (F,C)\in Pairs$$
$$\overrightarrow{ADJ}^{F}=(o_1,...,o_n),\ where o_i=\#(\theta_i,F,position=\textbf{adjacent})$$
$$\overrightarrow{DST}^{F}=(o_1,...,o_n),\ where o_i=\#(\theta_i,F,position=\textbf{distant})$$
$$\overrightarrow{ADJ}^{C}=(o_1,...,o_n),\ where o_i=\#(\theta_i,C,position=\textbf{adjacent})$$
$$\overrightarrow{DST}^{C}=(o_1,...,o_n),\ where o_i=\#(\theta_i,C,position=\textbf{distant})$$
$$Score_{F,C}=\frac{2}{3}\times\frac{\overrightarrow{ADJ}^{F}\bullet\overrightarrow{ADJ}^{C}}{\|\overrightarrow{ADJ}^{F}\|\times\|\overrightarrow{ADJ}^{C}\|}+\frac{1}{3}\times\frac{\overrightarrow{DST}^{F}\bullet\overrightarrow{DST}^{C}}{\|\overrightarrow{DST}^{F}\|\times\|\overrightarrow{DST}^{C}\|}$$

Figure 2. Mapping algorithm – refining step

The above frequency vectors are computed for FrameNet directly from the corpus of predicate-argument structure examples associated with each frame. The examples associated with the VerbNet lexicon are extracted from the PropBank corpus. In order to do this we apply a preprocessing step in which each label ARG0..N is replaced with its corresponding thematic role given the Intersective Levin class of the predicate. We assign the same roles to the adjuncts all over PropBank as they are general for all verb classes. The only exception is ARGM-DIR that can correspond to Source, Goal or Path. We assign different roles to this adjunct based on the prepositions. We ignore some adjuncts like ARGM-ADV or ARGM-DIS because they cannot bear a thematic role.

4.3 Mapping Results

We found that only 133 VerbNet classes have correspondents among FrameNet frames. Also, from the frames mapped with an automatic score smaller than 0.5 points almost a half did not match any of the existing VerbNet classes[3]. A summary of the results is depicted in Table 1. The first column contains the automatic score provided by the mapping algorithm when comparing frames with Intersective Levin classes. The second column contains the number of frames for each score interval. The third column contains the percentage of frames, per each score interval, that did not have a corresponding VerbNet class and finally the forth column contains the accuracy of the mapping algorithm.

Score	*No. of Frames*	*Not mapped*	*Correct*	*Overall Correct*
[0,0.5]	118	48.3%	82.5%	89.6%
(0.5,0.75]	69	0	84%	
(0.75,1]	72	0	100%	

Table 1. Results of the mapping algorithm

4.4 Discussion

In the literature, other studies compared the Levin classes to the FrameNet frames (Baker and Ruppenhofer, 2002). Their findings suggest that although the two set of clusters are roughly equivalent there are also several types of mistmaches: 1) Levin classes that are narrower than the corresponding frames, 2) Levin classes that are broader that the corresponding frames and 3) overlapping groupings. For our task, point 2 does not pose a problem. Points 1 and 3 however suggest that there are cases in which to one FrameNet frame corresponds more than one Levin class. By investigating such cases we noted that the mapping algorithm consistently assigns scores below 75% to cases that match problem 1 (two Levin classes inside one frame) and below 50% to cases that match problem 3 (more than two Levin classes inside one frame). Thus, in order to increase the accuracy of our results a first step should be to assign an

[3] The automatic mapping can be improved by manually assigning the FrameNet frames of the pairs that receive a score lower than 0.5.

Intersective Levin class to each of the verbs pertaining to frames with score lower than 0.75. Nevertheless the current results are encouraging as they show that the algorithm is achiving its purpose by successfully detecting syntactic incoherencies that can be subsequently corrected manually. Also, in the next section we will show that our current mapping achieves very good results, giving evidence for the effectivenes of the Levin class feature.

5 Experiments

In the previous section we have presented the algorithm for annotating the verb predicates of FrameNet with Intersective Levin classes. In order to show the effectiveness of this annotation and of the Intersective Levin class in general we have performed several experiments.

First, we trained (1) an ILC multiclassifier from FrameNet, (2) an ILC multiclassifier from PropBank and (3) a frame multiclassifier from FrameNet. We compared the results obtained when trying to classify the VerbNet class with the results obtained when classifying frame. We show that Intersective Levin classes are easier to detect than FrameNet frames.

Our second set of experiments regards the automatic labeling of FrameNet semantic roles on FrameNet corpus when using as features: gold frame, gold Intersective Levin class, automatically detected frame and automatically detected Intersective Levin class. We show that in all situations in which the VerbNet class feature is used, the accuracy loss, compared to the usage of the frame feature, is negligible. We thus show that the Intersective Levin class can successfully replace the frame feature for the task of semantic role labeling.

Another set of experiments regards the generalization property of the Intersective Levin class. We show the impact of this feature when very few training data is available and its evolution when adding more and more training examples. We again perform the experiments for: gold frame, gold Intersective Levin class, automatically detected frame and automatically detected Intersective Levin class.

Finally, we simulate the difficulty of free text by annotating PropBank with FrameNet semantic roles. We use PropBank because it is different from FrameNet from a domain point of view. This characteristic makes PropBank a difficult test bed for semantic role models trained on FrameNet.

In the following section we present the results obtained for each of the experiments mentioned above.

5.1 Experimental setup

The corpora available for the experiments were PropBank and FrameNet. PropBank contains about 54,900 sentences and gold parse trees. We used sections from 02 to 22 (52,172 sentences) to train the Intersective Levin class classifiers and section 23 (2,742 sentences) for testing purposes.

For the experiments on FrameNet corpus we extracted 58,384 sentences from the 319 frames that contain at least one verb annotation. There are 128,339 argument instances of 454 semantic roles. Only verbs are selected to be predicates in our evaluations. Moreover, as there is no fixed split between training and testing, we randomly selected 20% of sentences for testing and 80% for training. The sentences were processed using Charniak's parser (Charniak, 2000) to generate parse trees automatically.

For classification, we used the SVM-light-TK software available at `http://ai-nlp.info.uniroma2.it/moschitti` which encodes tree kernels in the SVM-light software (Joachims, 1999). The classification performance was evaluated using the F_1 measure for the single-argument classifiers and the accuracy for the multiclassifiers.

5.2 Automatic VerbNet vs. automatic FrameNet frame detection

In these experiments we classify Intersective Levin classes (ILC) on PropBank (PB) and FrameNet (FN) and frame on FrameNet. For the training stage we use SVMs with Tree Kernels.

The main idea of tree kernels is the modeling of a $K_T(T_1,T_2)$ function which computes the number of common substructures between two trees T_1 and T_2. Thus, we can train SVMs with structures drawn directly from the syntactic parse tree of the sentence.

The kernel that we employed in our experiments is based on the SCF structure devised in (Moschitti, 2004). We slightly modified SCF by adding the headwords of the arguments, useful for representing the selectional preferences.

For frame detection on FrameNet, we trained our classifier on 46,734 training instances and tested on 11,650 testing instances, obtaining an accuracy of 91.11%. For ILC detection the results are depicted in Table 2. The first six columns report the F_1 measure of some verb

class classifiers whereas the last column shows the global multiclassifier accuracy.

We note that ILC detection is performed better than frame detection on both FrameNet and PropBank. Also, the results obtained on ILC on PropBank are similar with the ones obtained on ILC on FrameNet. This suggests that the training corpus does not have a major influence. Also, the SCF-based tree kernel seems to be robust in what concerns the quality of the parse trees. The performance decay is very small on FrameNet that uses automatic parse trees with respect to PropBank that contains gold parse trees. These properties suggest that ILC are very suitable for free text.

	run-51.3.2	cooking-45.3	characterize-29.2	other_cos-45.4	say-37.7	correspond-36.1	Multiclassifier
PB #Train Instances	262	6	2,945	2,207	9,707	259	52,172
PB #Test Instances	5	5	134	149	608	20	2,742
PB Results	**75**	**33.33**	**96.3**	**97.24**	**100**	**88.89**	**92.96**
FN #Train Instances	5,381	138	765	721	1,860	557	46,734
FN #Test Instances	1,343	35	40	184	1,343	111	11,650
FN Results	**96.36**	**72.73**	**95.73**	**92.43**	**94.43**	**78.23**	**92.63**

Table 2 . F_1 and accuracy of the argument classifiers and the overall multiclassifier for Intersective Levin class

5.3 Automatic semantic role labeling on FrameNet

In the experiments involving semantic role labelling, we used a SVM with a polynomial kernel. We adopted the standard features developed for semantic role detection by Gildea and Jurafsky (see Section 2). Also, we considered some of the features designed by (Pradhan et al., 2004): *First and Last Word/POS in Constituent*, *Subcategorization*, *Head Word of Prepositional Phrases* and the *Syntactic Frame* feature from (Xue and Palmer, 2004). For the rest of the paper we will refer to these features as being literature features (LF). The results obtained when using the literature features alone or in conjunction with the gold frame feature, gold ILC, automatically detected frame feature and automatically detected ILC are depicted in Table 3. The first four columns report the F_1 measure of some role classifiers whereas the last column shows the global multiclassifier accuracy. The first row contains the number of training and testing instances and each of the other rows contains the performance obtained for different feature combinations. The results are reported for the labeling task as the argument-boundary detection task is not affected by the frame-like features (G&J).

We note that automatic frame results are very similar to automatic ILC results suggesting that ILC feature is a very good candidate for replacing the frame feature. Also, both automatic features are very effective, decreasing the error rate of 20%.

	Body_part	Crime	Degree	Agent	Multiclassifier
FN #Train Instances	1,511	39	765	6,441	102,724
FN #Test Instances	356	5	187	1,643	25,615
LF+Gold Frame	90.91	88.89	70.51	93.87	90.8
LF+Gold ILC	90.80	88.89	71.52	92.01	88.23
LF+Automatic Frame	**84.87**	**88.89**	**70.10**	**87.73**	**85.64**
LF+Automatic ILC	**85.08**	**88.89**	**69.62**	**87.74**	**84.45**
LF	79.76	75.00	64.17	80.82	80.99

Table 3. F_1 and accuracy of the argument classifiers and the overall multiclassifier for FrameNet semantic roles

5.4 Semantic role learning curve when using Intersective Levin classes

The next set of experiments show the impact of the ILC feature on semantic role labelling when few training data is available (Figure 3). As can be noted, the automatic ILC features (i.e. derived with classifers trained on FrameNet or PB) produce accuracy almost as good as the gold ILC one. Another observation is that the SRL classifiers are not saturated and more training examples would improve their accuracy.

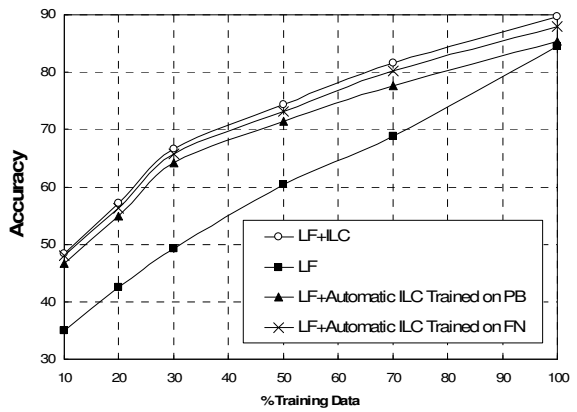


Figure 3. Semantic Role learning curve

5.5 Annotating PropBank with FrameNet semantic roles

To show that our approach can be suitable for semantic role free-text annotation, we have automatically classified PropBank sentences with the FrameNet semantic-role classifiers. In order to measure the quality of the annotation, we randomly selected 100 sentences and manually verified them. We measured the performance obtained with and without the automatic ILC feature. The sentences contained 189 arguments from which 35 were incorrect when ILC was used compared to 72 incorrect in the absence of this feature. This corresponds to an accuracy of 81% with Intersective Levin class versus 62% without it.

6 Conclusions

In this paper we have shown that the Intersective Levin class feature can successfully replace the FrameNet frame feature. By doing that we could interconnect FrameNet to VerbNet and PropBank obtaining better verb coverage and a more robust semantic parser. Our good results show that we have defined an effective framework which is a promising step toward the design of free-text semantic parsers.

In the future, we intend to measure the effectiveness of our system by testing on larger, more comprehensive corpora and without relying on any manual annotation.

Reference

Collin Baker and Josef Ruppenhofer. 2002. FrameNet's frames vs. Levin's verb classes. 28th Annual Meeting of the Berkeley Linguistics Society.

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the CoNLL-2005 Shared Task: Semantic Role Labeling. CONLL'05.

Eugene Charniak. 2000. A Maximum-Entropy-Inspired Parser. ANLP'00

Hoa Trang Dang, Karin Kipper, Martha Palmer and Joseph Rosenzweig. 1998. Investigating regular sense extensions based on Intersective Levin classes. Coling-ACL'98.

Charles Fillmore. 1968. The case for case. Universals in Linguistic Theory.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. CL Journal.

Christopher Johnson, Miriam Petruck, Collin Baker, Michael Ellsworth, Josef Ruppenhofer, and Charles Fillmore. 2003. FrameNet: Theory and Practice. Berkeley, California.

Paul Kingsbury, Martha Palmer. 2002. From TreeBank to PropBank. LREC'02.

Karin Kipper, Hoa Trang Dang and Martha Palmer. 2000. Class-based construction of a verb lexicon. AAAI'00.

Beth Levin. 1993. English Verb Classes and Alternations A Preliminary Investigation. Chicago: University of Chicago Press.

Kenneth Litkowski. 2004. Senseval-3 task automatic labeling of semantic roles. Senseval-3.

Paola Merlo and Suzanne Stevenson. 2001. Automatic verb classification based on statistical distribution of argument structure. CL Journal.

Alessandro Moschitti. 2004. A study on convolution kernel for shallow semantic parsing. ACL'04.

Sameer Pradhan, Kadri Hacioglu, Valeri Krugler, Wayne Ward, James H. Martin, and Daniel Jurafsky. 2004. Support vector learning for semantic argument classification. Machine Learning Journal.

Cynthia A. Thompson, Roger Levy, and Christopher Manning. 2003. A Generative Model for FrameNet Semantic Role Labeling. ECML'03.

Thorsten Joachims. 1999. Making large-scale SVM learning practical.. Advances in Kernel Methods - Support Vector Learning.

Nianwen Xue and Martha Palmer. 2004. Calibrating features for semantic role labeling. EMNLP'04.

Constructing a Rule Based Naming System for Thai Names Using the Concept of Ontologies

Chakkrit Snae
Department of Computer Science
Naresuan University
Phitsanulok, Thailand
chakkrits@nu.ac.th

Abstract

Names are important in many societies, even in technologically oriented ones which use ID systems or other ways to identify individual people. Names such as personal surnames are the most important as they are used in many processes, such as identifying of people, record linkage and for genealogical research as well. For Thai names this situation is a bit different in that in Thai the first names are most important. Even phone books and directories are sorted according to the first names. Here we present a system for constructing Thai names from basic syllables. Typically Thai names convey a meaning. For this we use an ontology of names to capture the meaning of the variants which are based on the Thai naming methodology and rules.

1 Introduction

Names are used for identifying persons, places, things and even ideas or concepts. Names serve for labelling of categories or classes and for individual items. They are properties of individuals which are of greater importance in most communities. In technological oriented societies such as modern Western the reference between names as a label and the person is not as obvious as in small tribal societies. This is especially true where names are stored within large information systems. This includes government, medical, educational and even commercial records which are kept about individuals. Names are the most important referrer to a person even if there are numbering systems like ID numbers because such systems are not universal. Names are often queried in a different way than they were entered. Names represent complex lexical structures which have to be handled systematically for data entry, storage and retrieval in order to get sufficient recall or precision the retrieval process in.

In this paper we present a first account of our findings on constructing Thai names with the help of an ontology of names as well as a working methodology for the naming process in Thai culture.

This paper is organized as follows: Section 2 contains a description of names and their elements. In Section 3 we outline the concept of ontology of names. In Section 4 we present the construction process and system for Thai names. We apply this system together with an ontology to construct names with an appropriate meaning. Section 5 shows the conclusions of our study and further work which has to be performed.

2 What is a Name?

Names for individuals are often called proper names, for humans sometimes also anthroponyms. Names for places are called toponyms, for bodies of water hydronyms, for ethnic groups ethnonyms, for metaphors metonyms and so on. Names are more than just strings of characters. Names show important information such as titles, gender, marital status, and even birthplace. For this names provide different elements (Section 2.2), which may differ between cultures.

2.1 Naming for Identity and Security

From the technical point of view we want to link and match as many names as possible with the correct individuals. If we deal with individuals of the same name, e.g. John Smith, we have to establish a second identifier at least. This can be – and is in many cases – a temporal element, like the date of birth, which is an individual and unchanging property of the person. Another way to circumvent the problem is to establish numbering systems, like ID numbers. Systems of numbers or other ciphers can be generated within individual organisations. It is not likely that the resulting ID numbers will be the same in different organisations. The numbering may have limitations as well, e.g. the individual health care setting (e.g. within a hospital or district) or, in principle, more widely (e.g. the National Health Service number). In the past, the National Health Service number in England and Wales had serious limitations as a matching variable, and it was not widely used on health-care records. With the allocation of the new ten-digit number throughout the NHS all this has been changed (Gill, 1997).

Although numbering systems are simple to implement they can lead to different errors in recording, transcripting, and keying. So we have to take into account methods which reduce these errors and facilitate good quality of data entry and retrieval. One such method uses a checking device such as check-digits (Wild, 1968, Hamming, 1986). When we are not able to use unique numbers or ciphers, natural matching variables are the person's name, date of birth, sex and perhaps other supplementary variables such as the address with postal code and place of birth, which are used in combination for matching. Recently, it has been suggested that this simple code could be extended for security critical places (e.g. airports, checkpoints etc.) with biometric marker information extracted from person identifier information e.g. fingerprints/iridograms.

2.2 Elements of Personal Names

The following table shows typical elements of personal names together with potential variations and sources of choices, e.g. dictionary of given names.

1. Initial and feudal names
- for male–Mr.
- for female
  - -single female –Ms
  - -married female – Mrs
- professional Occupation–Dr. , Prof. , ranking career
- honorary titles and feudal names for social status in society - Lord, Sir, Knight, Baron, etc.

2.First name, given name
- mixture of parent names
- monks
- naming system
- astrology
- family, e.g. great grandparents
- book/ dictionary of names
- using name that matches character e.g. actor/actress

3. Middle name
- grandparents or great grandparents
- parents

4. Surname
- inherited from family
- the king or Royal family
- parents surnames
- grandparents
- using name that matches character e.g. actor/actress

5. Nickname
- last syllable of given names, e.g. Chakkrit → Krit
- first syllable of given names, e.g. Robert→ Rob
- called by family
- called by friend / local community

6. Artist name and pseudonyms
- first name only, like Sasha
- surname only, like Chernyim (Thai comedian)
- only nickname, like Prince
- only abbreviation, like –ky
- for artists, monks, etc.

Figure 1: The elements of names

3 Ontology of Names?

The term ontology has been widely used in recent years in the field of Artificial Intelligence, computer and information science especially in domains such as, cooperative information systems, intelligent information integration, information retrieval and extraction, knowledge representation, and database management systems (Guarino, 1998, Andrade and Saltz, 1999, 2000). Many different definitions of the term are proposed. One of the most widely quoted and well-known definition of ontology is Gruber's (Gruber, 1993): An ontology is an explicit specification of a conceptualization.

The term is borrowed from philosophy, where an ontology is a systematic account of existence. Here in this paper we adopt the following definition: Ontology is the study or concern about what kinds of things exist - what entities or things are there in the universe (Blackburn, 1996). Our work on ontologies will comprise: a terminological component where we lay down the concepts and an assertional component (or Knowledge Base) which contains the individual instances (entities). The level of description will be taxonomies with hierarchically related terms and controlled vocabularies (thesaurus) with the help of semantic networks.

An ontology of names can be worked out in many different forms, but every ontology will include a dictionary, some definition of the terms

(semantics), and indications how they depend on each other, e.g. in hierarchies and semantic networks. For example, an ontology of names can be defined as what kinds of names exist, e.g. first name, surname, nickname, etc (Section 2.2). This typically comprises definitions of different names, the elements of names and their structures. In this section we show how an ontology of names can be captured and defined.

An ontology can also be used to establish the network of synonyms, e.g. using spelling norms to determine whether two names are the same/similar or not. For example, two names: Totie and Totiey can be defined based on assumption that they are the same as Totty. This attempts to tackle the seemingly irreducible conventions of surname. In compositional semantics let us consider the name "Gutensohn". This name will be used to illustrate the various semantic considerations in German naming. The name is a composition of the two strings Godith and Sohn, which have unambiguous, meaningful interpretations. The interpretation of Godith is god or good battle and Sohn is interpreted as a male child in relation to his parent. The composition Gutsohn, Gudzon, or in other cultures: Guditson, Godyeson and Godithson and Godison (Reaney and Wilson 1997).

We incorporate the different elements of personal names (Figure 1) into a semantic network (Figure 2) to illustrate how they associate with each other, e.g. with hierarchies.

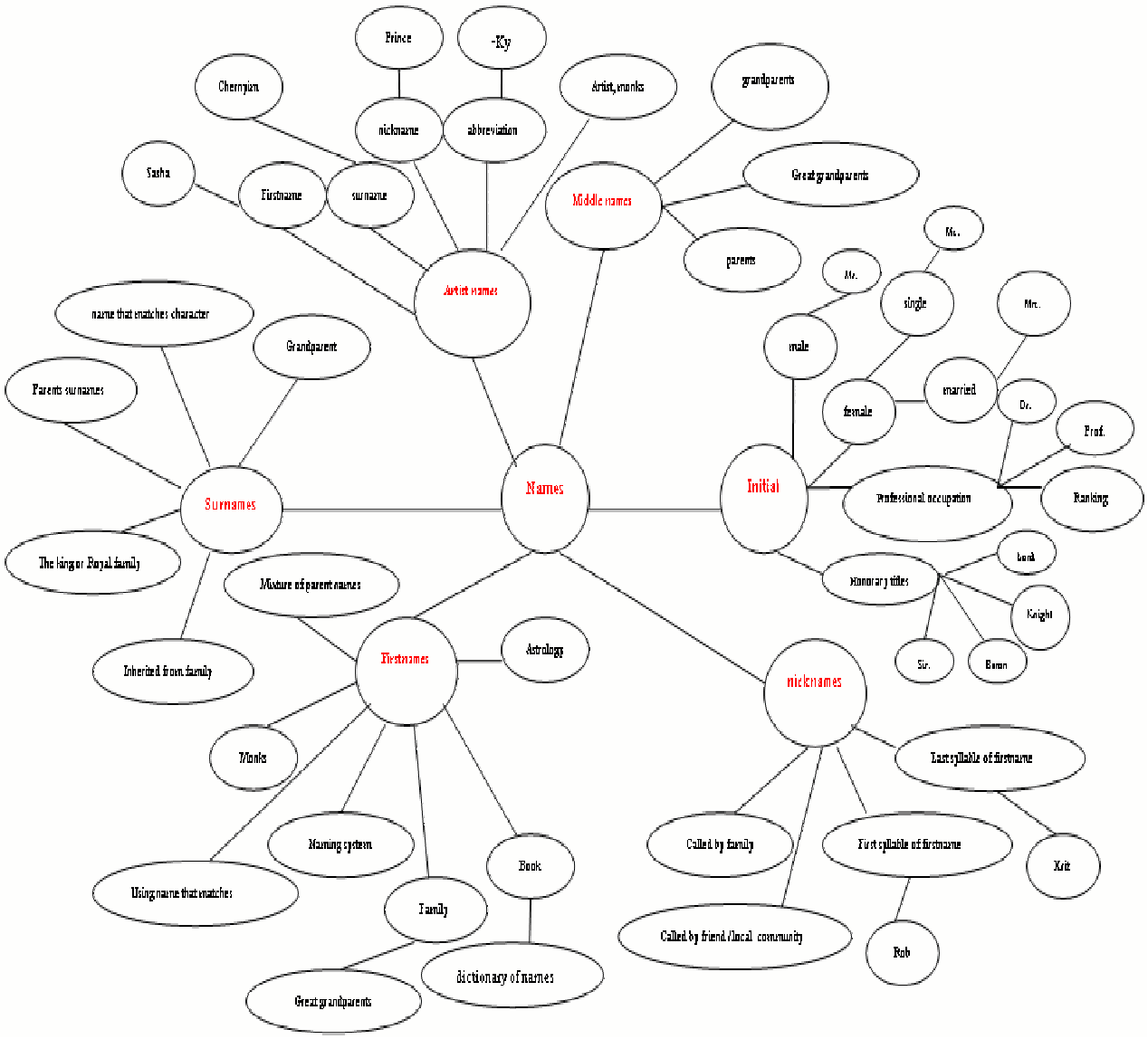


Figure 2: Representation of the names elements using semantic nets

Identifying and searching result in a list many names with variations and meanings. In order to find the correct person with a name we have to adopt ontologies of names, e.g. based on place of birth or relationship of people. The typical origins of surnames which can be a basis for ontologies of names can be classified as follows:
local surnames - surnames of relationship - surnames of occupation or office.

Local surnames, which are most widely used, stem from toponyms, we can call them toponymic. They reflect land owners, place of birth, or the center of life. For example, Richard de Tonebridge was named after his castle of Tonbridge, but he was also called Richard de Clara from the Suffolk Clare, which became his chief seat and the family's definitive surname. Also Richard de Hadestoke, a London alderman, had left Hadstock (Essex) and settled in London (Reaney and Wilson 1997). These local surnames derive (with occasional exceptions) from English, Scottish or French places (e.g. de, at, in). Toponymic Thai names are derived from Thai places and took originally a preposition na, for example, Prapas na Ranong is a person from a Southern province in Thailand called Ranong.

Surnames which come from family relation are often called patronymic, but we have to introduce a more elaborate term, because we encounter names from females and other relations than just father, such as Gilbert Fathevedsteppeson, Richard Hannebrothir, America Ibbotdoghter, and John Prestebruther.

Surnames of occupation and office refer to actual office holders like clergy names or state offices. Some of these, such as steward, constable, marshal, etc., became hereditary and gave rise to hereditary surnames, but the terms were also commonly used of lesser offices, whilst marshal was a common term for a farrier and such names frequently denoted the actual occupation. However, Nuns, Abbots, Priors, Monks and other clerical people were bound by vows of celibacy and thus did usually not have families which adopted their respective surname.

4 Rule Based Naming System for Thai Names

In this section we introduce the well known methodology for Thai naming process as well as how Thai names can be constructed using the basic Thai rules of forming syllables.

4.1 Custom Naming Process Using Thai Astrology

The way of naming can vary, e.g. naming by monks, grandparents. Since former times names are very important to people. Naming from the past to the present has been continuously developed and has developed a variety of patterns. Each pattern has it own rules depending on local places and the belief that has been developed until the present. The basic goal of naming is to provide a good fortune and progress during life. Most first names have a meaning. The widely used methodology of Thai naming process is briefly described in the following.

Principal naming using Thai astrology is widely used since the past. Because it uses the birth day to form the name. This is a belief that the individual has a set of 8 attributes called name of the angles referred to in Thai astrology. These attributes influence each person's livelihood, fortune, etc. The attributes refer to Servant >Age> Power> Honour> Property> Diligence> Patron> Misfortune. Each attribute has it own letters which can be used for constructing names.

4.2 Syllable Construction

Syllables are aggregated to names which sound good or aimed at good fortune according to the methodology mentioned above. As a consonant can not stand alone in Thai language and personal names we consider rules for vowels only. The order is:

Vowels can come first or can be followed by a first consonant, e.g. Ek

Vowels can follow a first consonant without a final consonant, e.g. Ka

Vowels that can not have final consonant, e.g. Tam, Tua

Vowels that need final consonant, e.g. Kak

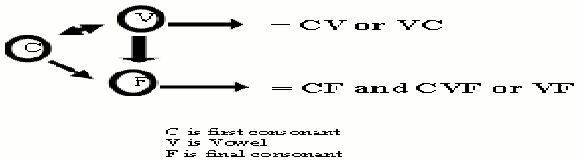


Figure 3: Forming of Thai syllables

Example of construction of Thai syllables using Thai Romanization 1.10 unicode (CU 2004) according to Figure 3: กา (Ka) = CV, เอก (Ek) = VC, กก (Kok) = CF, กั (Kak) = CVF, เอก (Ek) = VF.

Thai names are built from one or more syllables that may or may not have a meaning. There are syllables which alone do not mean much in particular, but when used as prefixes and suffixes can change the meaning of the syllables they precede or follow as stated below.

Prefixes: for example, (1) kob means "gather" while pra-kob means "put together" or "consist of", (2) cham means "remember" while pra-cham means "regularly".

Suffixes: for example, (1) ngarm means "beautiful" and num means "water". Add jai ("heart") to each and we have names like ngarm-jai and num-jai meaning "beautiful in the heart" and "generous" respectively.

In the following it is shown how to construct Thai names that convey a meaning with the help of ontologies. Syllables are built from consonants (either C or F, C being the first and F the final consonant) and vowels. A name consists of one or more syllables. One syllable can have a meaning of its own, which leads in case of two or more syllables in a name to more complex meanings.

The process of constructing names according to the naming rules and methodology begins with a leading consonant or vowel that can be the only letter in the name. If we continue to add more letters we come either to a valid name (a name which has a meaning) or to an invalid name (a name without a meaning). Invalid names will be discarded in such a way that the last letter will be replaced by another or will be added with more letters.

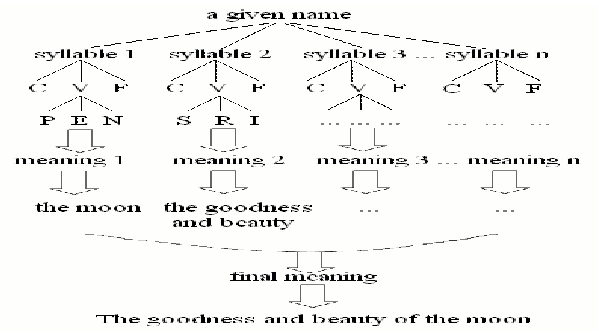


Figure 4: Representation and construction of Thai names

In Figure 4 it is shown that names comprise *n* syllables with a reasonable number of letters. The meanings of the syllables as well as of the name are found with the help of an ontology of names.

The meaning of the name for a girl Pensri in the example (see Figure 4) is “the goodness and beauty of the moon”.

How do we know which name belongs to a boy or a girl? There are several ideas to take into consideration when selecting a name for indicating the gender. Ontologies can help in deciding the name by capturing and indexing the meaning as is shown in Table 1.

Boys can be named by:

- taking names of leading or important male characters accepted as having lasting literary value, e.g. Chakkri (an actor from KhunChang KhunPhaen) and Suthat (an actor from Phra Aphai Mani).
- combining monosyllables evidently indicating the males sex, e.g. Chatri (a mighty man), Choetchai (a perfect man), and Danai (son).
- using adjectives indicating male qualities such as strength, and bravery, e.g. Watcharaphon (strength), woraphon (strength), and Kriangkrai (brevery) .
- using terms representing something strong, important or high, e.g. Suriya, Phassakorn, and Phanuphong are popular names. They all mean the sun.

Choosing names for girls is even more complicated. There are so many things to consider (Table 2). A girl can be named by:

- taking name of leading female characters from well-known literature: Phim Phi La Lai (an actress from Khun-Chang KhunPhaen) and Suwanmali (an actress from Phra Aphai Mani).
- combining monosyllables evidently indicating the females sex, e.g. Charuni (pretty girl), Kanyarat (pretty woman), Khanitda (younger sister),and Khwansuda (beloved daughter).
- using syllables indicating female qualities such as those having to do with looks and beauty, e.g. Pariyaphat (lovely) and Phichittra (beautiful).

- using syllables describing softness, gentleness and mildness, e.g. Pranit (gentle), Lamun (soft), and Sukhum (mild).
- using syllables describing scent, taste, e.g. Saowakhon (lovely smell, fragrant) and Parimon (fragrant), Mathurot (sweet) and Wasita (sweet and aromatic). On the other hand, unfavourable tastes like Khom (bitter) or Fad (sappy) are not used.
- using syllables which are names of ornaments and jewellery, e.g. Phatchara Walai (diamond bracelet), Rattana Wali (gem necklace), Phara (diamond), and Rachawadi (blue gems).
- using syllables which are names of flowers, flowering trees: Manlika (jasmine), Maliwan (jasmine), and Watsana (Charisma), all of which are names of flowers found in Thailand; also syllables which simply mean flower: Buppha, Phakamat, Butsabong, and Ladawan.

Thai name	Romanized	Meaning	Cluster
จักรี	Chakkri	an actor from KhunChang KhunPhaen	leading or important male
จักรกฤษณ์	Chak Krit	an actor from Unnarut	characters from well-known
สุทัศน์	Suthat	an actor from Phra Aphai Mani	literature
สัญชัย	San Chai	an actor from Wetsandon Chadok	
อินทรชิต	InThonChit	an actor from Rammakian	
ชาตรี	Chatri	a mighty *man*	
เชิดชาย	Choet Chai	a perfect *man*	
ยอดชาย	Yot Chai	a brilliant *man*	terms indicating the males
นพพล	Nopphon	a prestigious and vigorous *man*	sex
วิชรพล	Wi Chon Phon	a powerful *man*	
ดนัย	Danai	a *son*	
วัชรพล	Watchara Phon	*strength*	
วรพล	Woraphon	*strength*	
พลาธิป	Phla Thip	*strength*	adjectives indicating male
เกรียงไกร	Kriangkrai	*bravery*	qualities such as strength,
วีรภัทร	Wira Phat	*bravery*	and bravery
ธีรศานต์	Thira San	*brave* and quiet	
สุริยะ	Suriya	the *sun*	
ภาสกร	Phatsakon	the *sun*	
ภาณุพงศ์	Phanu Phong	tribe of the *sun*	terms representing
ราชันย์	Rachan	the *King*	something strong, important
ไกรสร	Kraison	lion *King*	or high
ฉัตรชัย	Chat Chai	triumph of the *king*	
ชนาธิป	Chana Thip	the *king*	

Table 1 Examples of Thai names with their meanings according to male gender.

Thai names	Romanized	Meaning	Cluster
นิพภาพร	Nirr Kasuri	an actress from Ramnakian	leading or important male characters from well-known literature
พิมพิลาไลย	Phim Phi La Lai	an actress from Khun Chang Khunphaen	
สุวรรณมาลี	Suwan Mali	an actress from Phra Aphai Mani	
กันทิมา	Kan Thi Ma	an actress from Phuchana Sip Thit	
ทิพเกสร	Thip Kesorn	an actress from Lak Sana Wong	
สาวิตรี	Sawittri	an actress from Sawittri	
จันทร์สุดา	Chan Suda	an actress from Hon Wichai Khawi	
จารุณี	Charu Ni	pretty *girl*	terms indicating the females sex
จิตรกัญญา	Chit Kanya	beautiful *girl*	
กัญญารัตน์	Kanya Rat	pretty *woman*	
ชนิษฐดา	Khanit Da	younger *sister*	
ขวัญนารี	Khwan Nari	younger *sister*	
ขวัญสุดา	Khwan Suda	beloved *daughter*	
จันทร์สุดา	Chan Suda	beautiful *daughter*	
ธิดาวรรณ	Thida Wan	beautiful *daughter*	
กันตพร	Kanta Phon	*pretty* and superb	syllables indicating female qualities such as pretty, lovely, and beautiful
ปิยาพร	Pi Ya Phon	*lovely* and goodness	
ปริยาภัทร	Pari Ya Phat	*lovely*	
พิมพ์วิไล	Phim Wilai	*beautiful*	
พิจิตรา	Phi Chittra	*beautiful*	
จิตรวิมล	Chit Wimon	very *beautiful*	
วิมพ์วิภา	Wim Wipha	*beautiful*	

Thai names	Romanized	Meaning	Cluster
ประณีต	Pranit	neat, *gentle*	syllables describing gentle, soft and mild
ละเมียด	Lamiat	*gentle*	
ละมุน	Lamun	*soft*, smooth, and *gentle*	
ละไม	Lamai	be nice, be *gentle*	
ละมอม	Lamom	*gentle* and *soft*	
สุขุม	Sukhum	*mild* and *soft*	
สุภาพ	Suphap	be polite, *gentle*	
มธุรส	Mathurot	*sweet*	syllables describing scent, taste
วาสิตา	Wa Si Ta	*sweet* and *aromatic*	
ปริมล	Pari Mon	*fragrant*	
ขันธารัตน์	Khan Tha Rat	*fragrant*	
เสาวคนธ์	Saowakhon	*fragrant*	
สุรภิน	Sura Phin	very *aromatic*	
พัชราลัย	Phatchara Walai	*diamond* bracelet	syllables representing names of ornaments and jewellery such as diamond and germs
พัชรา	Phat Ra	*diamond*	
พัชรีวรรณ	Phat Ri Wan	beautiful likes *diamond*	
มณีประภา	Mani Prapha	brilliant Gem	
รัตนโกศ	Rattana Kot	*gems* treasury	
รัตนาวลี	Rattana Wali	*gem* necklace	
วาวาวดี	Rochawadi	blue *gems*	
บุปผา	Buppha	*flower*	syllables which are names of flowers, flowering trees or meanings of flower
ผกามาศ	Phaka Mat	*flower*	
บุษบง	Butsabong	*flower*	
เบญจมาศ	Benchamat	a name of *Thai flowers*	
มัลลิกา	Mallika	a name of *Thai flowers* , Jasmine	
มะลิวัลย์	Mali Wan	a name of *Thai flowers*, Jasmine	
วาสนา	Watsana	a name of *Thai flowers*, Charisma	

Table 2 Examples of Thai names with their meanings and female gender.

This grouping process is used to build an ontology which has a (classified) database structure. This allows for speeding up the information retrieval process for the naming system.

4.3 Web Based Naming Expert System

Currently we are constructing and implementing a web-based naming expert system which offers two basic ways to come to “good” Thai names according to the first methodology mentioned above. The system will give us the letters for each date of birth. We use these letters to construct names based on the basic rules (see Figure 2). The user will be able to choose from a list of resulting possible names according to their respective meaning.

We use a dictionary database of more than 8.000 Thai names which contains not only the spelling, but also the meaning and correct pro-

nunciation. In situations where names follow the rules but do not have a meaning we compare the name with similar names in a dictionary database and check for similarity using a simple string matching scheme. Then the user can select the best name from the resulting list of names.

A second way to come to names is by using ontologies instead of basic spelling rules which are used according to the sex and date of birth. For this we check the different names against the date of birth by implementing an indexed database system of names from Thai dictionary for every day of a week.

5 Conclusion and Further Work

We have used Thai customs for the naming process, an ontology of names, and the basic rules for forming syllables in Thai to construct the rule based naming system. We want to extend the system using name matching algorithms to return the variants of names from a Thai dictionary with the relative probability of their similarity. To speed up this process we will use a database structure based on an ontology of names, e.g. by indexing names according to gender and meaning with the help of a Thai dictionary database. We will use a Romanized version of Thai names.

The advantage of this process would be an improvement of searching algorithms for Thai names in databases as well as in the internet. Here we will need name matching algorithms. The next step of development is to take into account different cultures.

Currently we are designing a system for multicultural name matching called NARESUAN-M^2 (NAme Recognition Expert System Using Automated Name Matching Methods). A primary objective here is to study how ontologies and algorithms can help in deciding which rules of naming system have to be implemented. This will also require an investigation into how ontologies that cover the different elements of names can be merged.

References

Henrique Andrade and Joel Saltz.1999. Towards a Knowledge Base Management System KBMS: An Ontology-Aware Database Management System DBMS, Proceedings of the 14th Brazilian Symposium on Databases, Florianopolis, Brazil.

Henrique Andrade and Joel Saltz. 2000. Query Optimization in Kess – An Ontology-Based KBMS. Proceedings of the 15th Brazilian Symposium on Databases (SBBD'2000). João Pessoa, Brazil.

Simon Blackburn. *1996. The Oxford Dictionary of Philosophy*, Oxford: OUP.

CU.2004. <http://www.arts.chula.ac.th/%7Eling/tts/> (Nov. 19, 2005)

Kerry Dematteis, Richard Lutz and Heather McCallum-Bayliss. 1998. Whose Name Is It: Names, Ownership and Databases. Originally written for: 1998 Annual Meeting American Name Society San Francisco, CA.

Leicester E. Gill. 1997. OX-LINK: The Oxford Medical Record Linkage System, Complex linkage made easy, Record Linkage Techniques. In: *Proceedings of an International Workshop and Exposition*, 15-33.

Thomas R. Gruber. 1993. *A Translation Approach to Portable Ontology Specification. Knowledge Acquisition*, Vol. 5, No. 2, 199-220.

Nicola Guarino. 1998. Formal Ontology and Information Systems. In: Nicola Guarino (Ed.): *Proceedings FOIS'98*, Amsterdam.

Richard W. Hamming. 1986. Coding and Information Theory. 2nd ed. Englewood Cliffs, NJ: Prentice Hall.

Daniel S. Jurafsky and James H. Martin. 2000. *Speech and Language Processing*, Prentice Hall.

Percy H. Reaney and R.M. Wilson. 1997. *A Dictionary of English Surnames*, Oxford: OUP.

W.G. Wild. 1968. The Theory of Modulus *N* Check Digit Systems. *The Computer Bulletin*, 12, 308-311.

Wikipedia. 2005. <http://en.wikipedia.org/wiki/Royal_Thai_General_System_of_Transcription> (Nov. 19, 2005)

Ian Winchester. 1973. On referring to ordinary historical persons. In: *Identifying People in the Past*. E.A. Wrigley (Ed.), 17-40.

Ian Winchester. 1970. The Linkage of Historical Records by Man and Computer: Techniques and Problems. *Journal of Interdisciplinary History*. 1, 107-124.

Author Index