# Natural Language Processing and Information Retrieval

## Part of Speech Tagging and Named Entity Recognition

**Alessandro Moschitti**
Department of information and communication technology
University of Trento
Email: moschitti@dit.unitn.it

# Parts of Speech

- 8 traditional parts of speech for IndoEuropean languages
  - Noun, verb, adjective, preposition, adverb, article, interjection, pronoun, conjunction, etc
  - Around for over 2000 years (Dionysius Thrax of Alexandria, c. 100 B.C.)
  - Called: parts-of-speech, lexical category, word classes, morphological classes, lexical tags, POS

# POS examples for English

| | | |
|---|---|---|
| N | noun | *chair, bandwidth, pacing* |
| V | verb | *study, debate, munch* |
| ADJ | adj | *purple, tall, ridiculous* |
| ADV | adverb | *unfortunately, slowly* |
| P | preposition | *of, by, to* |
| PRO | pronoun | *I, me, mine* |
| DET | determiner | *the, a, that, those* |
| CONJ | conjunction | *and, or* |

# Open vs. Closed classes

- Closed:
  - determiners: ***a, an, the***
  - pronouns: ***she, he, I***
  - prepositions: ***on, under, over, near, by, …***
- Open:
  - Nouns, Verbs, Adjectives, Adverbs.

# Open Class Words

- Nouns
  - Proper nouns (Penn, Philadelphia, Davidson)
    - English capitalizes these.
  - Common nouns (the rest).
  - Count nouns and mass nouns
    - Count: have plurals, get counted: goat/goats, one goat, two goats
    - Mass: don't get counted (snow, salt, communism) (*two snows)
- Adverbs: tend to modify things
  - Unfortunately, John walked home extremely slowly yesterday
  - Directional/locative adverbs (here,home, downhill)
  - Degree adverbs (extremely, very, somewhat)
  - Manner adverbs (slowly, slinkily, delicately)
- Verbs
  - In English, have morphological affixes (eat/eats/eaten)

# Closed Class Words

- Differ more from language to language than open class words
- Examples:
  - prepositions: *on, under, over, …*
  - particles: *up, down, on, off, …*
  - determiners: *a, an, the, …*
  - pronouns: *she, who, I, ..*
  - conjunctions: *and, but, or, …*
  - auxiliary verbs: *can, may should, …*
  - numerals: *one, two, three, third, …*

# Prepositions from CELEX

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| of | 540,085 | through | 14,964 | worth | 1,563 | pace | 12 |
| in | 331,235 | after | 13,670 | toward | 1,390 | nigh | 9 |
| for | 142,421 | between | 13,275 | plus | 750 | re | 4 |
| to | 125,691 | under | 9,525 | till | 686 | mid | 3 |
| with | 124,965 | per | 6,515 | amongst | 525 | o'er | 2 |
| on | 109,129 | among | 5,090 | via | 351 | but | 0 |
| at | 100,169 | within | 5,030 | amid | 222 | ere | 0 |
| by | 77,794 | towards | 4,700 | underneath | 164 | less | 0 |
| from | 74,843 | above | 3,056 | versus | 113 | midst | 0 |
| about | 38,428 | near | 2,026 | amidst | 67 | o' | 0 |
| than | 20,210 | off | 1,695 | sans | 20 | thru | 0 |
| over | 18,071 | past | 1,575 | circa | 14 | vice | 0 |

# Conjunctions

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| and | 514,946 | yet | 5,040 | considering | 174 | forasmuch as | 0 |
| that | 134,773 | since | 4,843 | lest | 131 | however | 0 |
| but | 96,889 | where | 3,952 | albeit | 104 | immediately | 0 |
| or | 76,563 | nor | 3,078 | providing | 96 | in as far as | 0 |
| as | 54,608 | once | 2,826 | whereupon | 85 | in so far as | 0 |
| if | 53,917 | unless | 2,205 | seeing | 63 | inasmuch as | 0 |
| when | 37,975 | why | 1,333 | directly | 26 | insomuch as | 0 |
| because | 23,626 | now | 1,290 | ere | 12 | insomuch that | 0 |
| so | 12,933 | neither | 1,120 | notwithstanding | 3 | like | 0 |
| before | 10,720 | whenever | 913 | according as | 0 | neither nor | 0 |
| though | 10,329 | whereas | 867 | as if | 0 | now that | 0 |
| than | 9,511 | except | 864 | as long as | 0 | only | 0 |
| while | 8,144 | till | 686 | as though | 0 | provided that | 0 |
| after | 7,042 | provided | 594 | both and | 0 | providing that | 0 |
| whether | 5,978 | whilst | 351 | but that | 0 | seeing as | 0 |
| for | 5,935 | suppose | 281 | but then | 0 | seeing as how | 0 |
| although | 5,424 | cos | 188 | but then again | 0 | seeing that | 0 |
| until | 5,072 | supposing | 185 | either or | 0 | without | 0 |

# Auxiliaries

| | | | | | |
|---|---|---|---|---|---|
| can | 70,930 | might | 5,580 | shouldn't | 858 |
| will | 69,206 | couldn't | 4,265 | mustn't | 332 |
| may | 25,802 | shall | 4,118 | 'll | 175 |
| would | 18,448 | wouldn't | 3,548 | needn't | 148 |
| should | 17,760 | won't | 3,100 | mightn't | 68 |
| must | 16,520 | 'd | 2,299 | oughtn't | 44 |
| need | 9,955 | ought | 1,845 | mayn't | 3 |
| can't | 6,375 | will | 862 | dare, have | ??? |

**Figure 5.5** English modal verbs from the CELEX on-line dictionary. Frequency counts are from the COBUILD 16 million word corpus.

# POS Tagging: Choosing a Tagset

- There are so many parts of speech, potential distinctions we can draw
- To do POS tagging, we need to choose a standard set of tags to work with
- Could pick very coarse tagsets
  - N, V, Adj, Adv.
- More commonly used set is finer grained, the "Penn TreeBank tagset", 45 tags
  - PRP$, WRB, WP$, VBG
- Even more fine-grained tagsets exist

# Penn TreeBank POS Tagset

| Tag | Description | Example | Tag | Description | Example |
|---|---|---|---|---|---|
| CC | coordin. conjunction | *and, but, or* | SYM | symbol | +,%, & |
| CD | cardinal number | *one, two, three* | TO | "to" | *to* |
| DT | determiner | *a, the* | UH | interjection | *ah, oops* |
| EX | existential 'there' | *there* | VB | verb, base form | *eat* |
| FW | foreign word | *mea culpa* | VBD | verb, past tense | *ate* |
| IN | preposition/sub-conj | *of, in, by* | VBG | verb, gerund | *eating* |
| JJ | adjective | *yellow* | VBN | verb, past participle | *eaten* |
| JJR | adj., comparative | *bigger* | VBP | verb, non-3sg pres | *eat* |
| JJS | adj., superlative | *wildest* | VBZ | verb, 3sg pres | *eats* |
| LS | list item marker | *1, 2, One* | WDT | wh-determiner | *which, that* |
| MD | modal | *can, should* | WP | wh-pronoun | *what, who* |
| NN | noun, sing. or mass | *llama* | WP$ | possessive wh- | *whose* |
| NNS | noun, plural | *llamas* | WRB | wh-adverb | *how, where* |
| NNP | proper noun, singular | *IBM* | $ | dollar sign | *$* |
| NNPS | proper noun, plural | *Carolinas* | # | pound sign | # |
| PDT | predeterminer | *all, both* | " | left quote | ' or " |
| POS | possessive ending | *'s* | " | right quote | ' or " |
| PRP | personal pronoun | *I, you, he* | ( | left parenthesis | [, (, {, < |
| PRP$ | possessive pronoun | *your, one's* | ) | right parenthesis | ], ), }, > |
| RB | adverb | *quickly, never* | , | comma | , |
| RBR | adverb, comparative | *faster* | . | sentence-final punc | . ! ? |
| RBS | adverb, superlative | *fastest* | : | mid-sentence punc | : ; ... – - |
| RP | particle | *up, off* | | | |

# Using the Penn Tagset

- The/DT grand/JJ jury/NN commmented/VBD on/IN a/DT number/NN of/IN other/JJ topics/NNS ./.

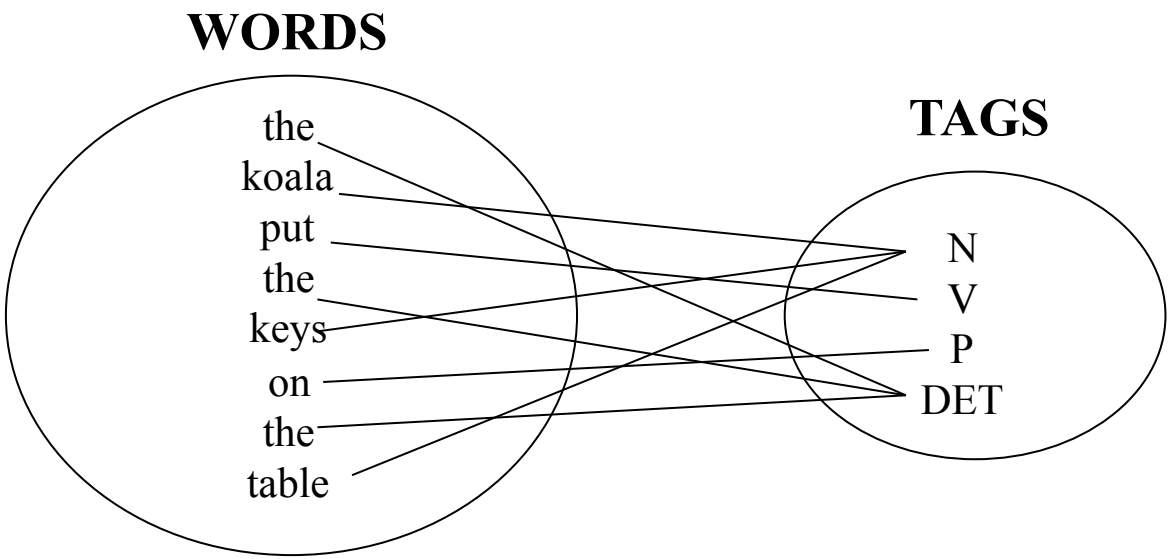
- Prepositions and subordinating conjunctions marked IN ("although/IN I/PRP..")
- Except the preposition/complementizer "to" is just marked "TO".

# Deciding on the correct part of speech can be difficult even for people

- Mrs/NNP Shaefer/NNP never/RB got/VBD around/RP to/TO joining/VBG
- All/DT we/PRP gotta/VBN do/VB is/VBZ go/VB around/IN the/DT corner/NN
- Chateau/NNP Petrus/NNP costs/VBZ around/RB 250/CD

# POS Tagging: Definition

- The process of assigning a part-of-speech or lexical class marker to each word in a corpus:



# POS Tagging example

| WORD | tag |
|---|---|
| **the** | **DET** |
| **koala** | **N** |
| **put** | **V** |
| **the** | **DET** |
| **keys** | **N** |
| **on** | **P** |
| **the** | **DET** |
| **table** | **N** |

# POS Tagging

- Words often have more than one POS: *back*
  - The *back* door = JJ
  - On my *back* = NN
  - Win the voters *back* = RB
  - Promised to *back* the bill = VB
- The POS tagging problem is to determine the POS tag for a particular instance of a word.

# How Hard is POS Tagging? Measuring Ambiguity

| | | 87-tag Original Brown | | 45-tag Treebank Brown | |
|---|---|---|---|---|---|
| **Unambiguous (1 tag)** | | **44,019** | | **38,857** | |
| **Ambiguous (2–7 tags)** | | **5,490** | | **8844** | |
| Details: | 2 tags | 4,967 | | 6,731 | |
| | 3 tags | 411 | | 1621 | |
| | 4 tags | 91 | | 357 | |
| | 5 tags | 17 | | 90 | |
| | 6 tags | 2 | (*well*, *beat*) | 32 | |
| | 7 tags | 2 | (*still*, *down*) | 6 | (*well*, *set*, *round*, *open*, *fit*, *down*) |
| | 8 tags | | | 4 | (*'s*, *half*, *back*, *a*) |
| | 9 tags | | | 3 | (*that*, *more*, *in*) |

# How difficult is POS tagging?

- About 11% of the word types in the Brown corpus are ambiguous with regard to part of speech
- But they tend to be very common words
- 40% of the word tokens are ambiguous

# Rule-Based Tagging

- Start with a dictionary
- Assign all possible tags to words from the dictionary
- Write rules by hand to selectively remove tags
- Leaving the correct tag for each word.

# Start With a Dictionary

- she: PRP
- promised: VBN,VBD
- to TO
- back: VB, JJ, RB, NN
- the: DT
- bill: NN, VB

- Etc… for the ~100,000 words of English with more than 1 tag

# Assign Every Possible Tag and apply rules

| | | | | | |
|---|---|---|---|---|---|
| | | | NN | | |
| | | | RB | | |
| | VBN | JJ | | VB | |
| PRP | VBD | TO | VB | DT | NN |
| **She** | **promised** | **to** | **back** | **the** | **bill** |

# Simple Statistical Approaches: Idea 1

**Simply assign each word its most likely POS.**

**Success rate: 91%!**

| Word | POS listings in Brown | | |
|---|---|---|---|
| heat | noun/89 | **verb/5** | |
| oil | **noun/87** | | |
| in | **prep/20731** | noun/1 | adv/462 |
| a | **det/22943** | noun/50 | noun-proper/30 |
| large | **adj/354** | noun/2 | adv/5 |
| pot | **noun/27** | | |

# Simple Statistical Approaches: Idea 2

For a string of words

$$W = w_1 w_2 w_3 \ldots w_n$$

find the string of POS tags

$$T = t_1\, t_2\, t_3 \ldots t_n$$

which maximizes P(T|W)

- i.e., the probability of tag string T given that the word string was W
- i.e., that W was tagged T

# Again, The Sparse Data Problem …

A Simple, Impossible Approach to Compute P(T|W):

Count up instances of the string "heat oil in a large pot" in the training corpus, and pick the most common tag assignment to the string..

# A Practical Statistical Tagger

**By Bayes' Rule:**

$$\mathbf{P(T|W)} = \frac{P(W|T) * P(T)}{P(W)}$$

so to maximize $P(T|W)$, need to maximize $P(W|T) * P(T)$.

**To compute P(T):** By the chain rule,

$$P(T) = P(t_1) * P(t_2|t_1) * P(t_3|t_1 t_2) * \ldots * P(t_n|t_1 \ldots t_{n-1})$$

# A Practical Statistical Tagger II

But we can't accurately estimate more than tag bigrams or so…

Again, we change to a model that we CAN estimate:

**A Markov Assumption:** $\mathrm{P}(\mathrm{t_i}|\mathrm{t_1} \ldots \mathrm{t_{i-1}}) = \mathrm{P}(\mathrm{t_i}|\mathrm{t_{i-1}})$

By which

$$P(T) = P(t_1) * P(t_2|t_1) * P(t_3|t_2) * \ldots * P(t_n|t_{n-1})$$

# A Practical Statistical Tagger III

**To compute $\mathbf{P(W|T)}$, similarly assume**

$$\mathbf{P(w_i|t_1 \ldots t_n) = P(w_i|t_i)}$$

By which

$$\boldsymbol{P(W|T) = P(w_1|t_1) * P(w_2|t_2) * \ldots * P(w_n|t_n)}$$

$$P(T) * P(W|T) = P(t_1) * P(t_2|t_1) * P(t_3|t_2) * \ldots * P(t_n|t_{n-1}) * P(w_1|t_1) * P(w_2|t_2) * \ldots * P(w_n|t_n)$$

So, for a given string W = $w_1w_2w_3\ldots w_n$, the tagger needs to *find the string of tags T which maximizes*

# Training and Performance

- To estimate the parameters of this model, given an annotated training corpus:

**To estimate $\mathbf{P(t_i|t_{i-1})}$:**

$$\frac{Count(t_{i-1}t_i)}{Count(t_{i-1})}$$

**To estimate $\mathbf{P(w_i|t_i)}$:**

$$\frac{Count(\ w_i \text{ tagged } t_i)}{Count(\text{ all words tagged } t_i)}$$

- Because many of these counts are small, *smoothing* is necessary for best results…
- Such taggers typically achieve about 95-96% correct tagging, **for tag sets of 40-80 tags.**

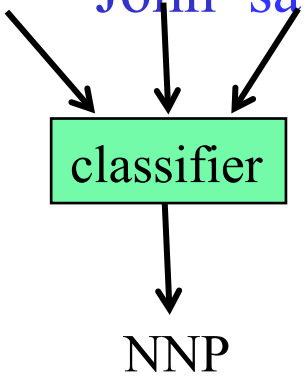
# Assigning tags to unseen words

- Pretend that each unknown word is ambiguous among all possible tags, with equal probability
- Assume that the probability distribution of tags over unknown words is like the distribution of tags over words seen only once
- Morphological clues
- Combination

# Sequence Labeling as Classification

- Classify each token independently but use as input features, information about the surrounding tokens (sliding window).

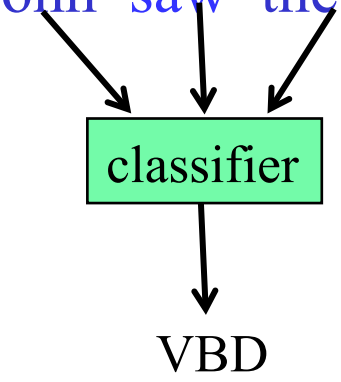
John saw the saw and decided to take it to the table.




# Sequence Labeling as Classification

- Classify each token independently but use as input features, information about the surrounding tokens (sliding window).

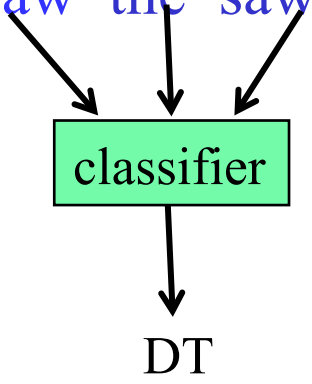
John saw the saw and decided to take it to the table.



# Sequence Labeling as Classification

- Classify each token independently but use as input features, information about the surrounding tokens (sliding window).

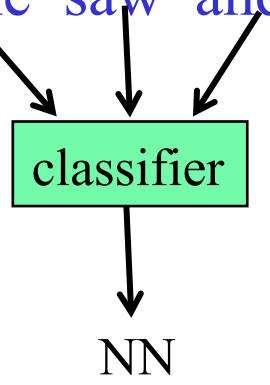
John saw the saw and decided to take it to the table.




# Sequence Labeling as Classification

- Classify each token independently but use as input features, information about the surrounding tokens (sliding window).

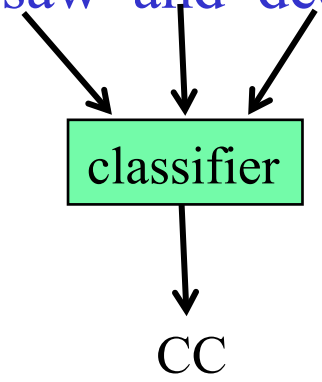
John saw the saw and decided to take it to the table.



# Sequence Labeling as Classification

- Classify each token independently but use as input features, information about the surrounding tokens (sliding window).

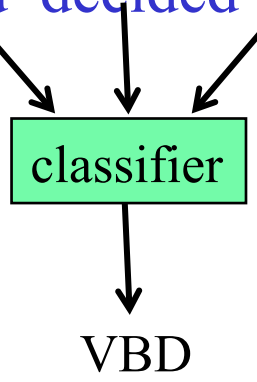
John saw the saw and decided to take it to the table.



# Sequence Labeling as Classification

- Classify each token independently but use as input features, information about the surrounding tokens (sliding window).

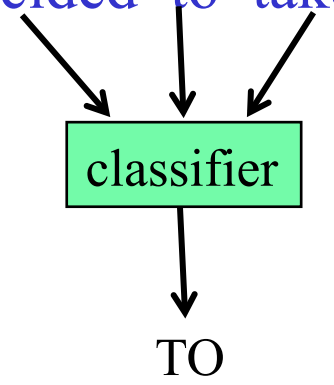
John saw the saw and decided to take it to the table.




# Sequence Labeling as Classification

- Classify each token independently but use as input features, information about the surrounding tokens (sliding window).

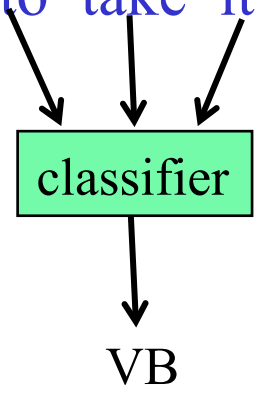
John saw the saw and decided to take it to the table.




# Sequence Labeling as Classification

- Classify each token independently but use as input features, information about the surrounding tokens (sliding window).

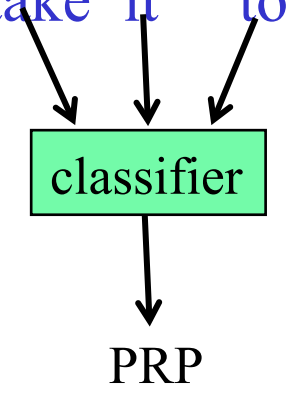
John saw the saw and decided to take it to the table.




# Sequence Labeling as Classification

- Classify each token independently but use as input features, information about the surrounding tokens (sliding window).

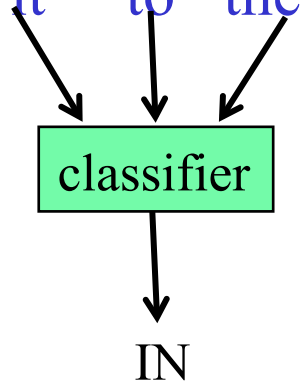
John saw the saw and decided to take it to the table.



# Sequence Labeling as Classification

- Classify each token independently but use as input features, information about the surrounding tokens (sliding window).

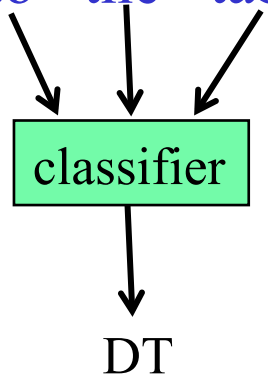
John saw the saw and decided to take it to the table.




# Sequence Labeling as Classification

- Classify each token independently but use as input features, information about the surrounding tokens (sliding window).

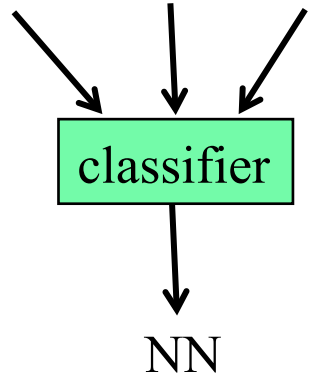
John saw the saw and decided to take it to the table.




# Sequence Labeling as Classification

- Classify each token independently but use as input features, information about the surrounding tokens (sliding window).

John saw the saw and decided to take it to the table.




# Sequence Labeling as Classification Using Outputs as Inputs

- Better input features are usually the categories of the surrounding tokens, but these are not available yet.
- Can use category of either the preceding or succeeding tokens by going forward or back and using previous output.

# SVMs for tagging

- http://www.lsi.upc.edu/~nlp/SVMTool/SVMTool.v1.4.ps
- We can use SVMs in a similar way
- We can use a window around the word
- 97.16 % on WSJ

# SVMs for tagging

| | |
|---|---|
| word unigrams | $w_{-3}, w_{-2}, w_{-1}, w_0, w_{+1}, w_{+2}, w_{+3}$ |
| word bigrams | $(w_{-2}, w_{-1}), (w_{-1}, w_{+1}), (w_{-1}, w_0), (w_0, w_{+1}), (w_{+1}, w_{+2})$ |
| word trigrams | $(w_{-3}, w_{-2}, w_{-1}), (w_{-2}, w_{-1}, w_0), (w_{-2}, w_{-1}, w_{+1}),$ $(w_{-1}, w_0, w_{+1}), (w_{-1}, w_{+1}, w_{+2}), (w_0, w_{+1}, w_{+2})$ |
| POS unigrams | $p_{-3}, p_{-2}, p_{-1}$ |
| POS bigrams | $(p_{-2}, p_{-1}), (p_{-1}, a_{+1}), (a_{+1}, a_{+2})$ |
| POS trigrams | $(p_{-3}, p_{-2}, p_{-1}), (p_{-2}, p_{-1}, a_{+1}), (p_{-1}, a_{+1}, a_{+2})$ |
| ambiguity classes | $a_0, a_1, a_2, a_3$ |
| maybe's | $m_0, m_1, m_2, m_3$ |
| prefixes | $s_1$, $s_1 s_2$, $s_1 s_2 s_3$, $s_1 s_2 s_3 s_4$ |
| suffixes | $s_n$, $s_{n-1} s_n$, $s_{n-2} s_{n-1} s_n$ $s_{n-3} s_{n-2} s_{n-1} s_n$ |
| binary word-form features | intial_Upper_Case, all_Upper_Case, no-initial_Capital_Letter(s), all_Lower_Case, contains_(period/number/hyphen ...) |
| word length | integer |
| Sentence info | last_word ('.', '?', '!') |

from Jimenez & Marquez

# An example of Features

Pierre NNP Vinken NNP , , 61 CD years NNS
old JJ , , will ?? join VB the DT board NN
as IN a DT nonexecutive JJ director NN
Nov. NNP 29 CD . .

*w-3*:years *w-2*:old
*w-1*:, *w0*:will *w1*:join
*w2*:the *w3*:board *Swn*:.

*w-2,-1*:old~, *w-1,1*:,~join
*w-1,0*:,~will *w0,1*:will~join
*w1,2*:join~the

*w-1,0,1*:,~will~join *w-1,1,2*:,~join~the
*w-2,-1,0*:old~,~will *w-2,-1,1*:old~,~join
*w0,1,2*:will~join~the

*p-3*:NNS *p-2*:JJ *p-1*:, *p-2,-1*:JJ~, *p-1,1*:,~VB_VBP
*p1,2*:VB_VBP~DT *p-2,-1,1*:JJ~,~VB_VBP *p-1,1,2*:,~VB_VBP~DT
*k0*:MD~NN *k1*:VB~VBP *k2*:DT *k3*:NN *s0~MD*:1
*s0~NN*:1 *s1~VB*:1 *s1~VBP*:1 *s2~DT*:1 *s3~NN*:1

# No sequence modeling

- Can do surprisingly well just looking at a word by itself:
  - Word: the: the → DT
  - Lowercased word: Importantly: importantly → RB
  - Prefixes: unfathomable: un- → JJ
  - Suffixes: Surprisingly: -ly → RB
  - Capitalization: Meridian: CAP → NNP
  - Word shapes: 35-year: d-x → JJ
- Then build a maxent (or whatever) model to predict tag
- Maxent P(t|w): 93.7% / 82.6%

# Evaluation

- So once you have you POS tagger running how do you evaluate it?
  - Overall error rate with respect to a gold-standard test set.
  - Error rates on particular tags
  - Error rates on particular words
  - Tag confusions...

# Evaluation

- The result is compared with a manually coded "Gold Standard"
  - Typically accuracy reaches 96-97%
  - This may be compared with result for a baseline tagger (one that uses no context).
- Important: 100% is impossible even for human annotators.

# Error Analysis

| | IN | JJ | NN | NNP | RB | VBD | VBN |
|---|---|---|---|---|---|---|---|
| **IN** | — | .2 | | | .7 | | |
| **JJ** | .2 | — | **3.3** | 2.1 | 1.7 | .2 | **2.7** |
| **NN** | | **8.7** | — | | | | .2 |
| **NNP** | .2 | **3.3** | **4.1** | — | .2 | | |
| **RB** | **2.2** | 2.0 | .5 | | — | | |
| **VBD** | | .3 | .5 | | | — | **4.4** |
| **VBN** | | **2.8** | | | | **2.6** | — |

- See what errors are causing problems
  - Noun (NN) vs ProperNoun (NNP) vs Adj (JJ)
  - Past tense verb form (VBD) vs Participle (VBN) vs Adjective (JJ)

# Named Entity Recognition

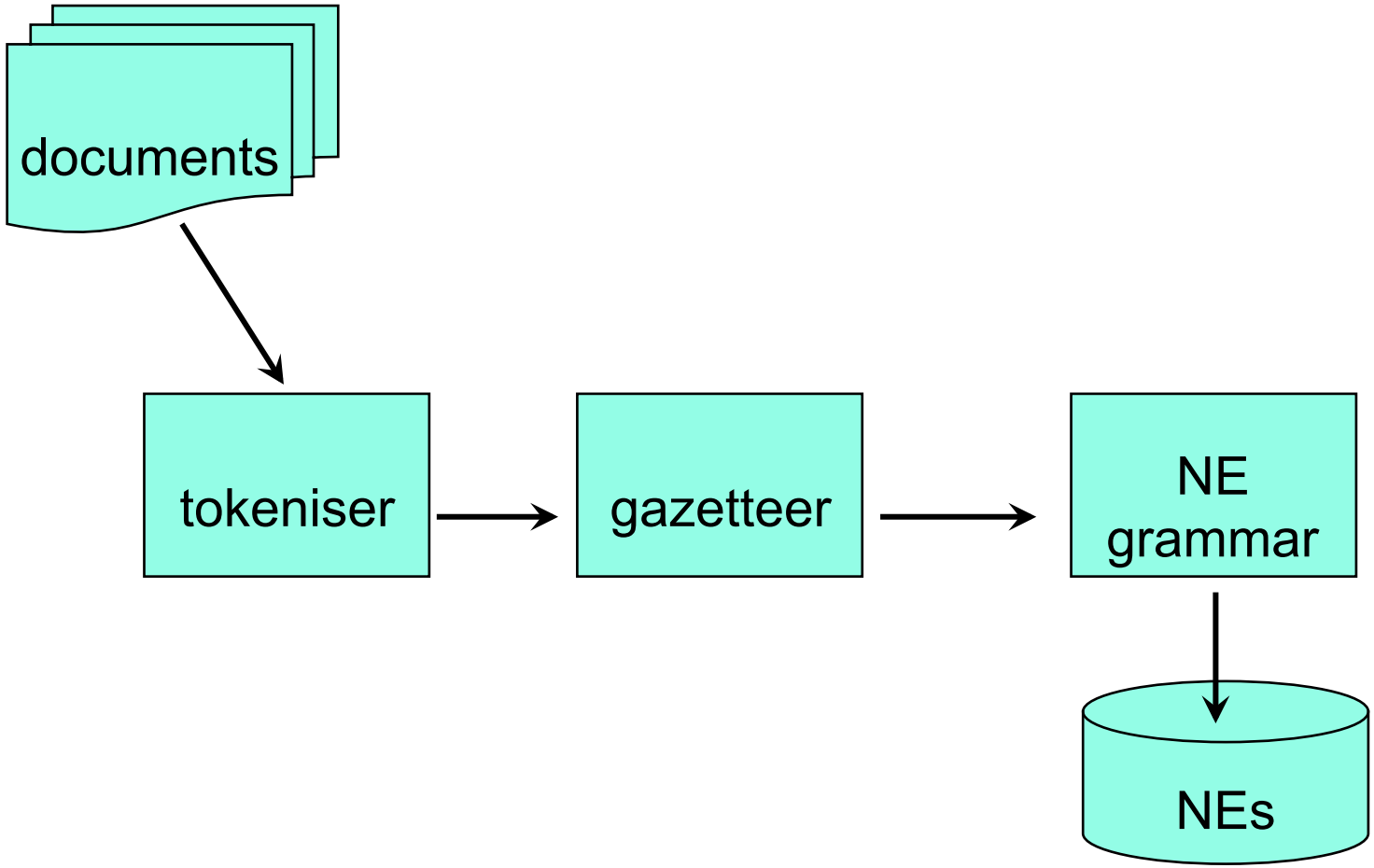
# Linguistically Difficult Problem

- NE involves **identification** of *proper names* in texts, and **classification** into a set of predefined categories of interest.
- Three universally accepted categories: **person**, **location** and **organisation**
- Other common tasks: recognition of **date/time expressions**, **measures** (percent, money, weight etc), email addresses etc.
- Other domain-specific entities: names of drugs, medical conditions, names of ships, bibliographic references etc.

# Problems in NE Task Definition

- Category definitions are intuitively quite clear, but there are many grey areas.
- Many of these grey area are caused by **metonymy**.
  - Organisation vs. Location : "**England** won the World Cup" vs. "The World Cup took place in **England**".
  - Company vs. Artefact: "shares in **MTV**" vs. "watching **MTV**"
  - Location vs. Organisation: "she met him at **Heathrow**" vs. "the **Heathrow** authorities"

# NE System Architecture



# Approach con't

- Again Text Categorization
- N-grams in a window centered on the NER
- Features similar to POS-tagging
  - **Gazetteer**
  - Capitalize
  - Beginning of the sentence
  - Is it all capitalized

# Approach con't

- NE task in two parts:
  - Recognising the entity boundaries
  - Classifying the entities in the NE categories
- Tokens in text are often coded with the IOB scheme
  - O – outside, B-XXX – first word in NE, I-XXX – all other words in NE
  - Easy to convert to/from inline MUC-style markup
  - Argentina B-LOC
    played O
    with O
    Del B-PER
    Bosque I-PER

# Feature types

- Word-level features
- List lookup features
- Document & corpus features

# Word-level features

**Table 1: Word-level features**

| Features | Examples |
|---|---|
| Case | - Starts with a capital letter<br>- Word is all uppercased<br>- The word is mixed case (e.g., ProSys, eBay) |
| Punctuation | - Ends with period, has internal period (e.g., St., I.B.M.)<br>- Internal apostrophe, hyphen or ampersand (e.g., O'Connor) |
| Digit | - Digit pattern (*see section 3.1.1*)<br>- Cardinal and Ordinal<br>- Roman number<br>- Word with digits (e.g., W3C, 3M) |
| Character | - Possessive mark, first person pronoun<br>- Greek letters |
| Morphology | - Prefix, suffix, singular version, stem<br>- Common ending (*see section 3.1.2*) |
| Part-of-speech | - proper name, verb, noun, foreign word |
| Function | - Alpha, non-alpha, n-gram (*see section 3.1.3*)<br>- lowercase, uppercase version<br>- pattern, summarized pattern (*see section 3.1.4*)<br>- token length, phrase length |

# List lookup features

| Features | Examples |
|---|---|
| General list | - General dictionary (see section 3.2.1)<br>- Stop words (function words)<br>- Capitalized nouns (e.g., January, Monday)<br>- Common abbreviations |
| List of entities | - Organization, government, airline, educational<br>- First name, last name, celebrity<br>- Astral body, continent, country, state, city |
| List of entity cues | - Typical words in organization (see 3.2.2)<br>- Person title, name prefix, post-nominal letters<br>- Location typical word, cardinal point |

Exact match vs. flexible match

Stems (remove inflectional and derivational suffixes)

Lemmas (remove inflectional suffixes only)

Small lexical variations (small edit distance)

Normalize words to their Soundex codes

# Document and corpus features

**Table 3: Features from documents.**

| Features | Examples |
|---|---|
| Multiple occurrences | - Other entities in the context<br>- Uppercased and lowercased occurrences (see 3.3.1)<br>- Anaphora, coreference (see 3.3.2) |
| Local syntax | - Enumeration, apposition<br>- Position in sentence, in paragraph, and in document |
| Meta information | - Uri, Email header, XML section, (see section 3.3.3)<br>- Bulleted/numbered lists, tables, figures |
| Corpus frequency | - Word and phrase frequency<br>- Co-occurrences<br>- Multiword unit permanency (see 3.3.4) |

# Examples of uses of document and corpus features

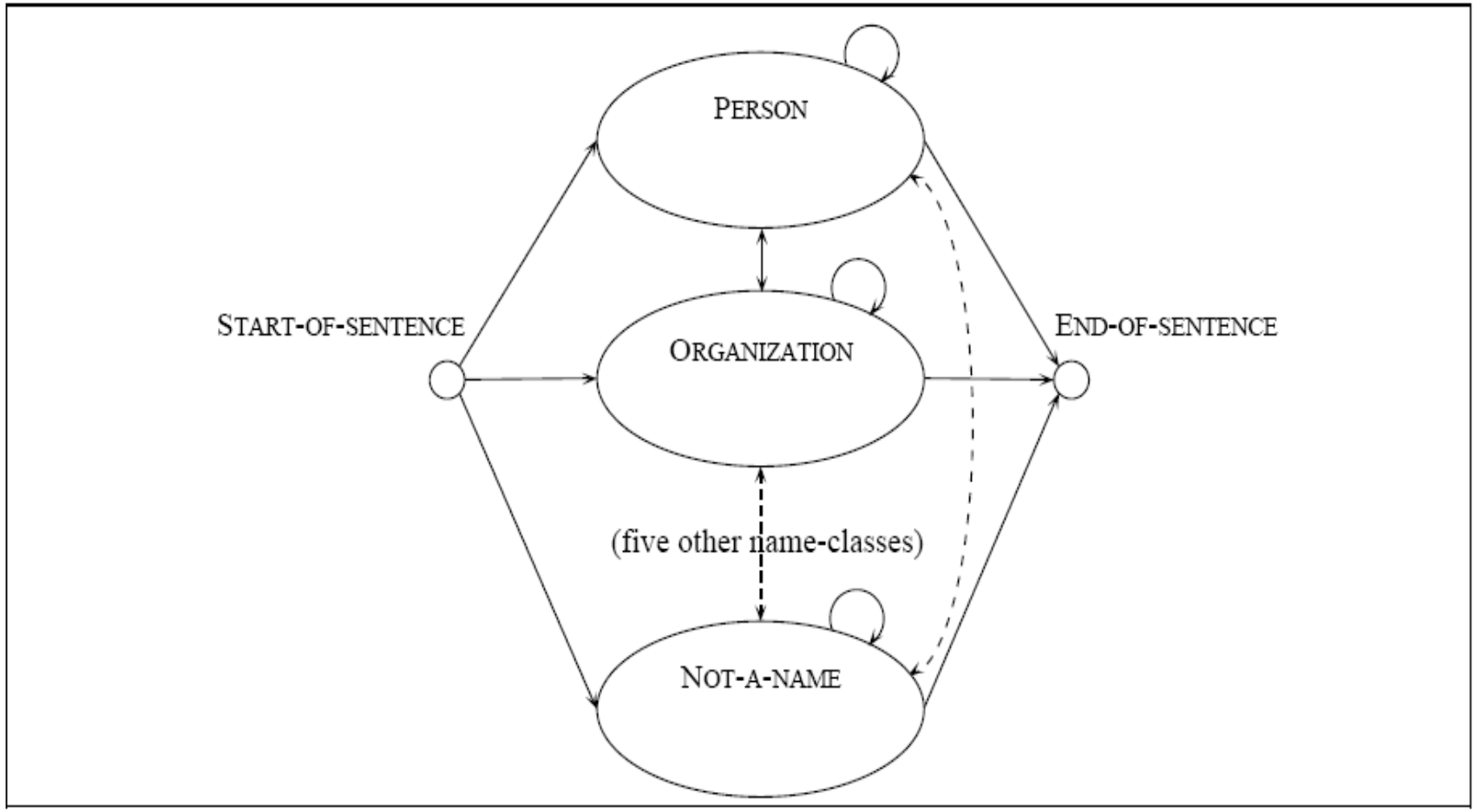
- Meta-information (e.g. names in email headers)
- Multiword entities that do not contain rare lowercase words of a relatively long size are candidate NEs
- Frequency of a word (e.g. Life) divided by its frequency in case insensitive form

# NER

- Description
- Performance

# Name Entity Recognition

- IndentiFinder (Bikel et al, 1999)
- Given a set of Named Entities (NE)
  - PERSON, ORGANIZATION, LOCATION, MONEY, DATE, TIME, PERCENT
- Predict NEs of a sentence with Hidden Markov models
  - $P(NC \mid NC_{-1}, w)$
  - $P(w \mid w_{-1} NC_{-1})$


PERSON
START-OF-SENTENCE
ORGANIZATION
END-OF-SENTENCE
(five other name-classes)
NOT-A-NAME

# Probability of "Mr. John eats."

Pr(NOT-A-NAME | START-OF-SENTENCE, "+end+") *
Pr("Mr." | NOT-A-NAME, START-OF-SENTENCE) *
Pr(+end+ | "Mr.", NOT-A-NAME) *
Pr(PERSON | NOT-A-NAME, "Mr.") *
Pr("Jones" | PERSON, NOT-A-NAME) *
Pr(+end+ | "Jones", PERSON) *
Pr(NOT-A-NAME | PERSON, "Jones") *
Pr("eats" | NOT-A-NAME, PERSON) *
Pr("." | "eats", NOT-A-NAME) *
Pr(+end+ | ".", NOT-A-NAME) *
Pr(END-OF-SENTENCE | NOT-A-NAME, ".")

# Other characteristics

- Probabilities are learned from annotated documents
- Features
- Levels of back-off
- Unknown models

| Word Feature | Example Text | Intuition |
|---|---|---|
| twoDigitNum | 90 | Two-digit year |
| fourDigitNum | 1990 | Four digit year |
| containsDigitAndAlpha | A8956-67 | Product code |
| containsDigitAndDash | 09-96 | Date |
| containsDigitAndSlash | 11/9/89 | Date |
| containsDigitAndComma | 23,000.00 | Monetary amount |
| containsDigitAndPeriod | 1.00 | Monetary amount, percentage |
| otherNum | 456789 | Other number |
| allCaps | BBN | Organization |
| capPeriod | M. | Person name initial |
| firstWord | *first word of sentence* | No useful capitalization information |
| initCap | Sally | Capitalized word |
| lowerCase | can | Uncapitalized word |
| other | , | Punctuation marks, all other words |

# Back-off levels

| Name-class Bigrams | First-word Bigrams | Non–first-word Bigrams |
|---|---|---|
| $\Pr(NC \mid NC_{-1}, w_{-1})$ | $\Pr(\langle w, f\rangle_{first} \mid NC, NC_{-1})$ | $\Pr(\langle w, f\rangle \mid \langle w, f\rangle_{-1}, NC)$ |
| ⋮ | ⋮ | ⋮ |
| $\Pr(NC \mid NC_{-1})$ | $\Pr(\langle w, f\rangle \mid \langle +begin+, \texttt{other}\rangle, NC)$ | $\Pr(\langle w, f\rangle \mid NC)$ |
| ⋮ | ⋮ | ⋮ |
| $\Pr(NC)$ | $\Pr(\langle w, f\rangle \mid NC)$ | $\Pr(w \mid NC) \cdot \Pr(f \mid NC)$ |
| ⋮ | ⋮ | ⋮ |
| $\frac{1}{\text{number of name - classes}}$ | $\Pr(w \mid NC) \cdot \Pr(f \mid NC)$ | $\frac{1}{\lvert V\rvert} \cdot \frac{1}{\text{number of word features}}$ |
| | ⋮ | |
| | $\frac{1}{\lvert V\rvert} \cdot \frac{1}{\text{number of word features}}$ | |

# Current Status

- Software Implementation
  - Learner and classifier in C++
  - Classifier in Java (to be integrated in Chaos)
- Named Entity Recognizer for English
  - Trained on MUC-6 data
- Named Entity Recognizer for Italian
  - Trained our annotated documents

# Contributions on Italian Versions

- Annotation of 220 documents from “La Repubblica”
- Modification of some features, e.g. “date”
- Accent treatments, e.g Cinecittà

# English Results

| | ACT | REC | PRE |
|---|---|---|---|
| SUBTASK SCORES | | | |
| enamex | | | |
| organization | 454 | 85 | 84 |
| person | 381 | 90 | 88 |
| location | 126 | 94 | 82 |
| timex | | | |
| date | 109 | 95 | 97 |
| time | 0 | 0 | 0 |
| numex | | | |
| money | 87 | 97 | 85 |
| percent | 26 | 94 | 62 |

**Precision = 91%**
**Recall = 87%**
**F1 = 88.61**

# Italian Corpus from “La Repubblica”

## Training data

| Class | Subtype | N° | Total |
|---|---|---|---|
| ENAMEX | Person | 1825 | 3886 |
| | Organization | 769 | |
| | Location | 1292 | |
| TIMEX | Date | 511 | 613 |
| | Time | 102 | |
| NUMEX | Money | 105 | 223 |
| | Percent | 118 | |

# Italian Corpus from “La Repubblica”

## Test data

| Class | Subtype | N° | Total |
|---|---|---|---|
| ENAMEX | Person | 333 | 537 |
| | Organization | 129 | |
| | Location | 75 | |
| TIMEX | Date | 45 | 48 |
| | Time | 3 | |
| NUMEX | Money | 5 | 13 |
| | Percent | 8 | |

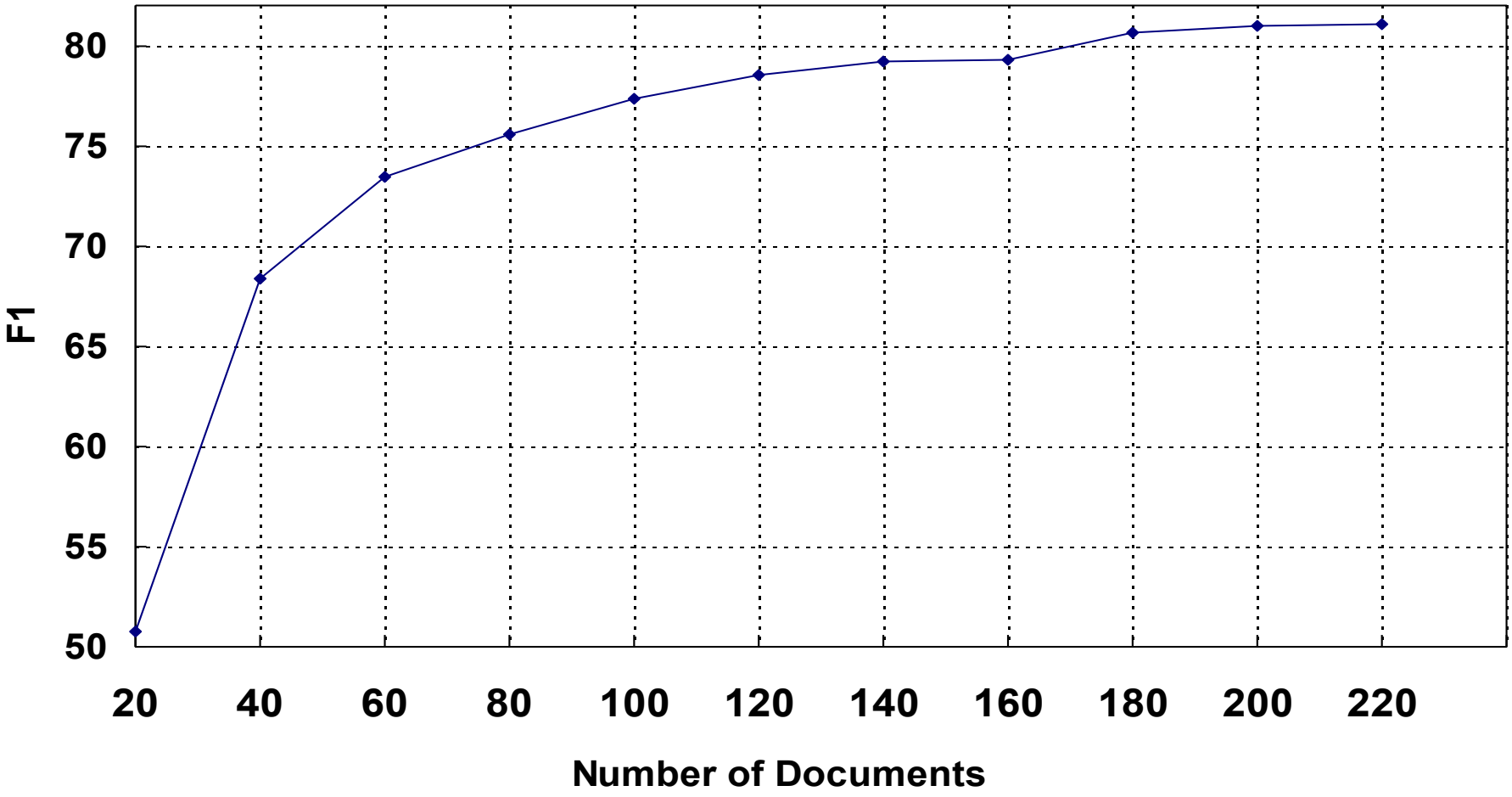
# Results of the Italian NER

- 11-fold cross validation (confidence at 99%)

| | Basic Model | +Modified Features | +Accent treatment |
|---|---|---|---|
| **Average F1** | **77.98±2.5** | **79.08±2.5** | **79.75±2.5** |

- Results on the development set 88.7 %
  - We acted only on improving annotation

# Learning Curve

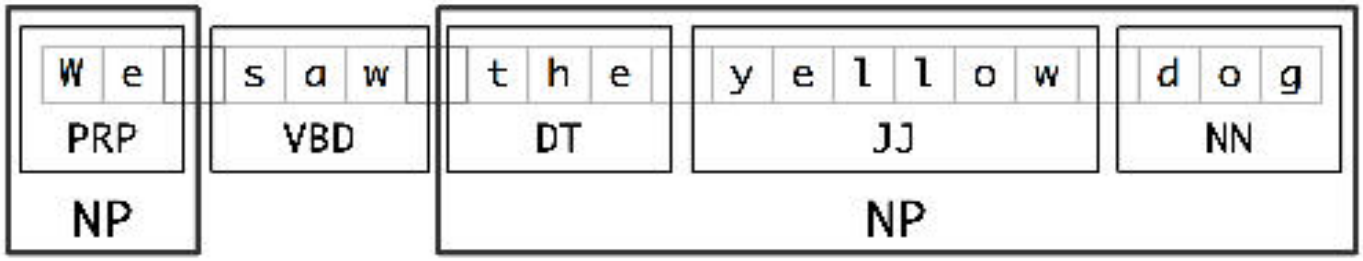


# Applications of NER

- Yellow pages with local search capabilities
- Monitoring trends and sentiment in textual social media
- Interactions between genes and cells in biology and genetics

# Chunking

- Chunking useful for entity recognition
- Segment and label multi-token sequences




- Each of these larger boxes is called a chunk

# Chunking

- The CoNLL 2000 corpus contains 270k words of Wall Street Journal text, annotated with part-of-speech tags and chunk tags.

```
>>> from nltk.corpus import conll2000
>>> print conll2000.chunked_sents('train.txt')[99]
(S
  (PP Over/IN)
  (NP a/DT cup/NN)
  (PP of/IN)
  (NP coffee/NN)
  ,/,
  (NP Mr./NNP Stone/NNP)
  (VP told/VBD)
  (NP his/PRP$ story/NN)
  ./.)
```

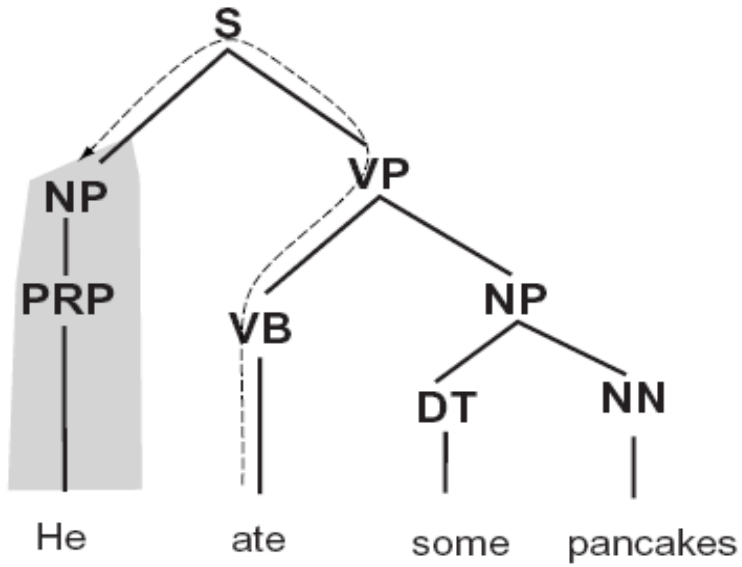
Three chunk types in CoNLL 2000:

- NP chunks
- VP chunks
- PP chunks

[NP He ] [VP reckons ] [NP the current account deficit ] [VP will narrow ]
[PP to ] [NP only # 1.8 billion ] [PP in ] [NP September ]



# No Path Feature available




| *Path* | *Description* |
|---|---|
| VB↑VP↓PP | PP argument/adjunct |
| VB↑VP↑S↓NP | subject |
| VB↑VP↓NP | object |
| VB↑VP↑VP↑S↓NP | subject (embedded VP) |
| VB↑VP↓ADVP | adverbial adjunct |
| NN↑NP↑NP↓PP | prepositional complement of noun |



• From Dan Kein's CS 288 slides (UC Berkeley)