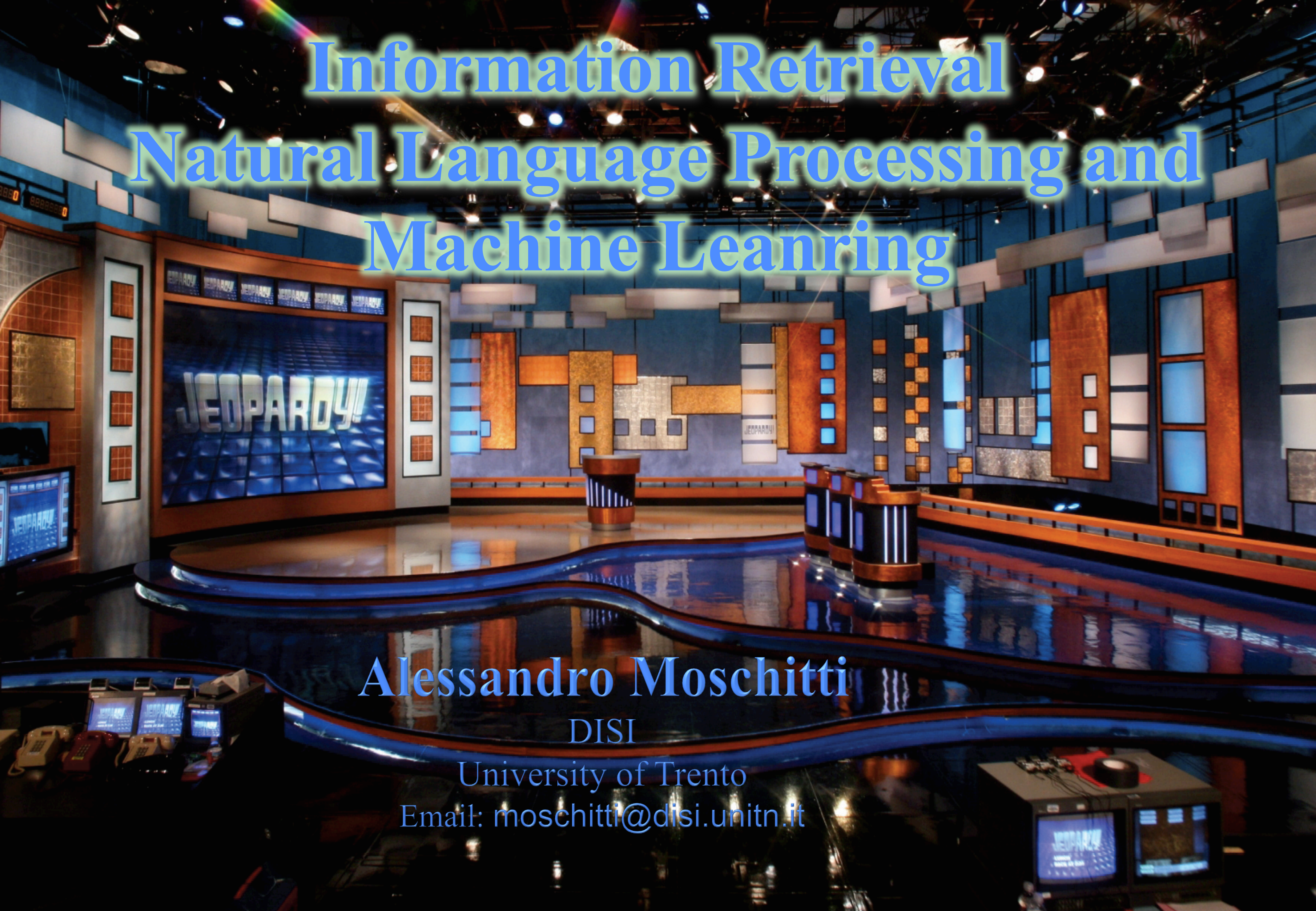


Information Retrieval Natural Language Processing and Machine Leanring

Alessandro Moschitti

DISI

University of Trento

Email: moschitti@disi.unitn.it

Advanced Natural Language Processing and Information Retrieval

PART I: Essential Notions of Information Retrieval and Machine Learning

Alessandro Moschitti

Department of Computer Science and Information Engineering
University of Trento
Email: moschitti@disi.unitn.it

Outline

- Motivation
  - Question Answering vs. Search Engines
- Information Retrieval Techniques
  - Search Engines
  - Vector Space Model
  - Feature Vectors and Feature Selection
  - Text Categorization
  - Measures
- Machine Learning Methods
  - Classification
  - Ranking
  - Regression (logistic)

Outline

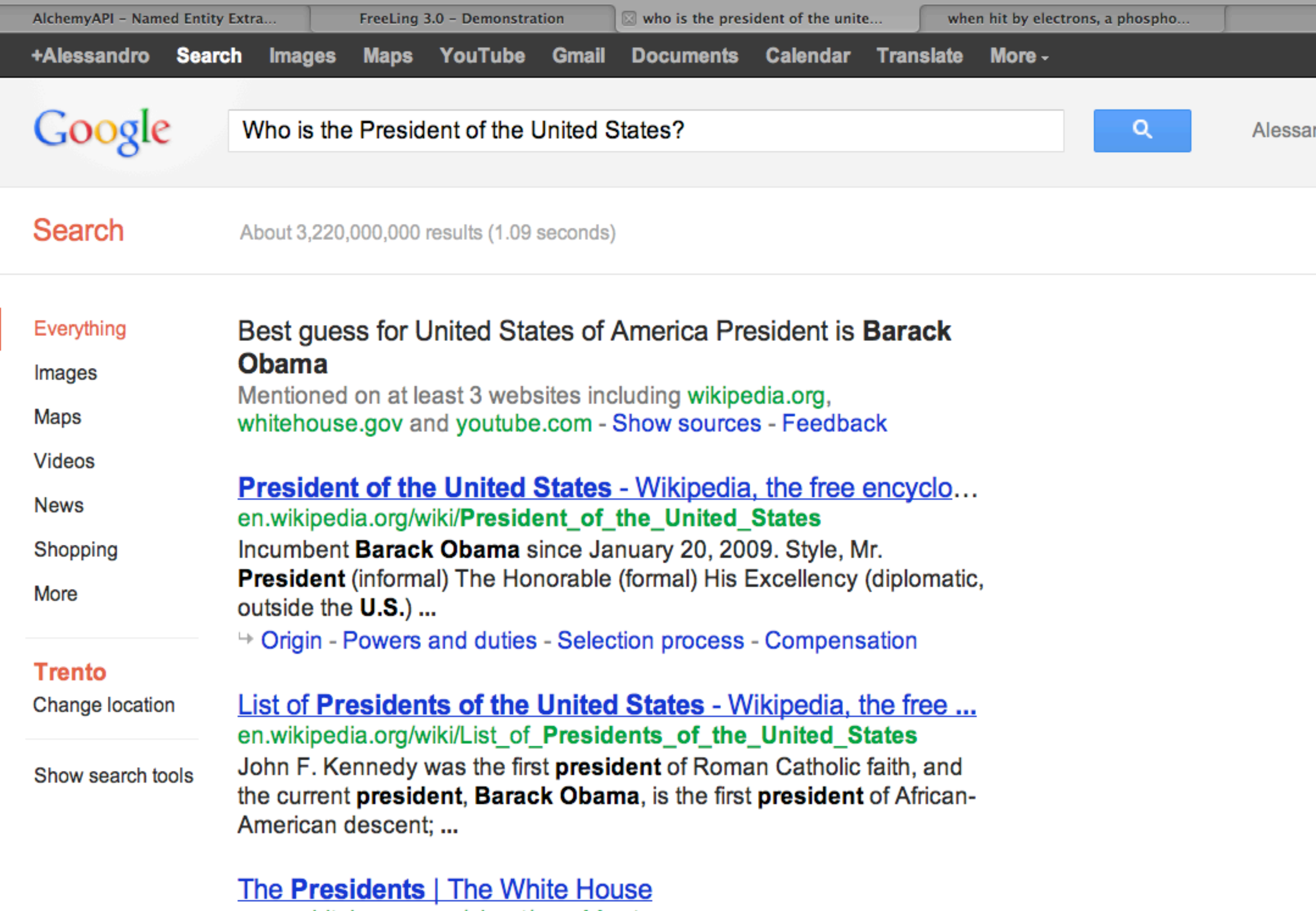
- Natural Language tools and techniques
  - Lemmatization
  - POS tagging
  - NER + gazetteer look up
  - Dependency and Constituency trees
  - Predicate Argument Structure
- Question Answering Pipeline
  - Similarity for supporting answers
  - QA tasks (open, restricted, factoid, non-factoid)

Motivations

- Approach to automatic Question Answering Systems
  1. Extract query keywords from the question
  2. Retrieve candidate passages containing such keywords (or synonyms)
  3. Select the most promising passage by means of query and answer similarity
- For example
  - Who is the President of the United States?

(Yes) The president of the United States is Barack Obama

(no) Glenn F. Tilton is President of the United Airlines


AlchemyAPI – Named Entity Extra...
FreeLing 3.0 – Demonstration
who is the president of the unite...
when hit by electrons, a phospho...
+Alessandro
Search
Images
Maps
YouTube
Gmail
Documents
Calendar
Translate
More
Google
Who is the President of the United States?
Alessar
Search
About 3,220,000,000 results (1.09 seconds)
Everything
Images
Maps
Videos
News
Shopping
More
Trento
Change location
Show search tools
Best guess for United States of America President is Barack Obama
Mentioned on at least 3 websites including wikipedia.org, whitehouse.gov and youtube.com - Show sources - Feedback
President of the United States - Wikipedia, the free encyclo...
en.wikipedia.org/wiki/President_of_the_United_States
Incumbent Barack Obama since January 20, 2009. Style, Mr. President (informal) The Honorable (formal) His Excellency (diplomatic, outside the U.S.) ...
Origin - Powers and duties - Selection process - Compensation
List of Presidents of the United States - Wikipedia, the free ...
en.wikipedia.org/wiki/List_of_Presidents_of_the_United_States
John F. Kennedy was the first president of Roman Catholic faith, and the current president, Barack Obama, is the first president of African-American descent; ...
The Presidents | The White House

Motivations

- TREC has taught that this model is to weak
- Consider a more complex task, i.e. a Jeopardy cue
- *When hit by electrons, a phosphor gives off electromagnetic energy in this form*
  - Solutions: ***photons/light***
- What are the most similar fragments retrieved by a search engine?

ps Maps News Shopping Gmail more

When hit by electrons, a phosphor gives off electromagnetic energy

About 194,000 results (0.22 seconds)

Advan

Cathode-Ray Tube - body, used, chemical, characteristics, **form ...**

Sep 6, 2010 ... In order to **form** the **electron** beam into the correct shape, ... The actual conversion of electrical **energy** to light **energy** takes place on the ... For example, the **phosphor** known as yttrium oxide **gives off** a red glow ... complete explanation of electrostatic and **electromagnetic** focusing in the crt ...

www.scienceclarified.com › Ca-Ch - Cached - Similar

Beta particle - Wikipedia, the free encyclopedia

Beta particles are high-**energy**, high-speed **electrons** or positrons emitted by certain ... The beta particles emitted are a **form** of ionizing radiation also known as beta rays. ... by **electromagnetic** interactions and may **give off** bremsstrahlung x-rays. ... The well-known 'betalight' contains tritium and a **phosphor**. ...

en.wikipedia.org/wiki/Beta_particle - Cached - Similar

luminescence: Definition from Answers.com

Included on the **electromagnetic** spectrum are radio waves and microwaves; ... Though the Sun sends its **energy** to Earth in the **form** of light and heat from the .... Thanks to the **phosphor**, a fluorescent lamp **gives off** much more light than an ... The tube itself is coated

Motivations (2)

- This shows that:
  - Lexical similarity is not enough
  - Structure is required
- What kind of structures do we need?
- How to carry out structural similarity?

Information Retrieval Techniques

Indexing Unstructured Text

- Which plays of Shakespeare contain the words ***Brutus*** *AND* ***Caesar*** but *NOT* ***Calpurnia***?
- One could grep all of Shakespeare's plays for ***Brutus*** and ***Caesar,*** then strip out lines containing ***Calpurnia***?
  - Slow (for large corpora)
  - *NOT* ***Calpurnia*** is non-trivial
  - Other operations (e.g., find the word ***Romans*** near ***countrymen***) not feasible
  - Ranked retrieval (best documents to return)

Term-document incidence

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	1	1	0	0	0	1
Brutus	1	1	0	1	0	0
Caesar	1	1	0	1	1	1
Calpurnia	0	1	0	0	0	0
Cleopatra	1	0	0	0	0	0
mercy	1	0	1	1	1	1
worser	1	0	1	1	1	0

1 if play contains word, 0 otherwise

Brutus** AND **Caesar** but NOT **Calpurnia

Incidence vectors

- So we have a 0/1 vector for each term.
- To answer query: take the vectors for ***Brutus, Caesar*** and ***Calpurnia*** (complemented) $\Rightarrow$ bitwise *AND*.
- 110100 *AND* 110111 *AND* 101111 = 100100.

Term-document incidence

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Brutus, Caesar* and *not Calpurnia	1	0	0	1	0	0
Antony	1	1	0	0	0	1
Brutus	1	1	0	1	0	0
Caesar	1	1	0	1	1	1
Calpurnia	0	1	0	0	0	0
Cleopatra	1	0	0	0	0	0
mercy	1	0	1	1	1	1
worser	1	0	1	1	1	0

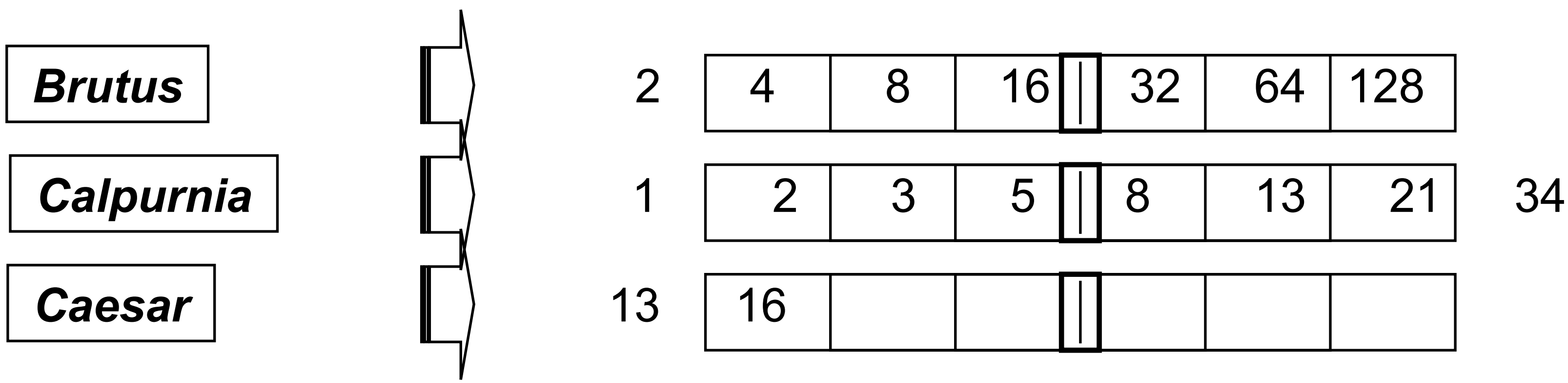
1 if play contains word, 0 otherwise

Brutus *AND* ***Caesar*** but *NOT* ***Calpurnia***

Inverted index

For each term *T*, we must store a list of all documents that contain *T*.

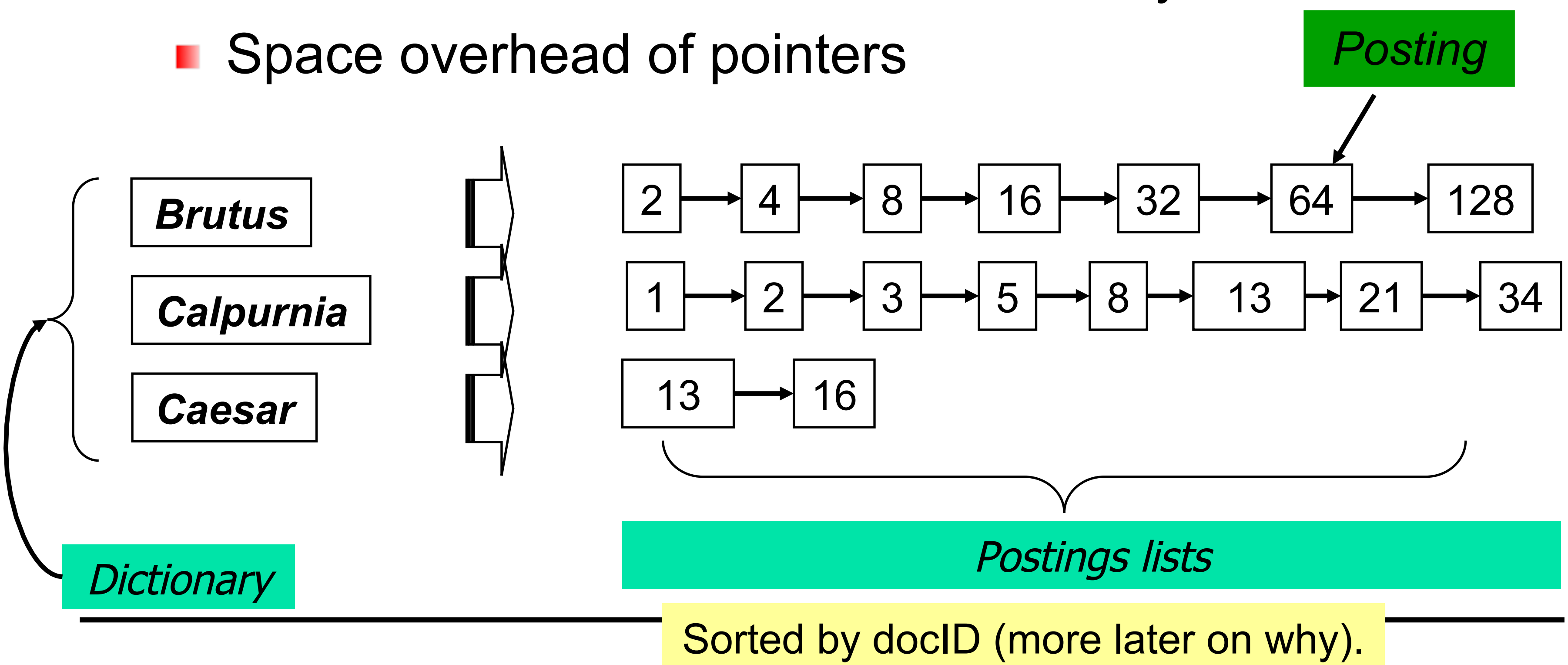
Do we use an array or a list for this?



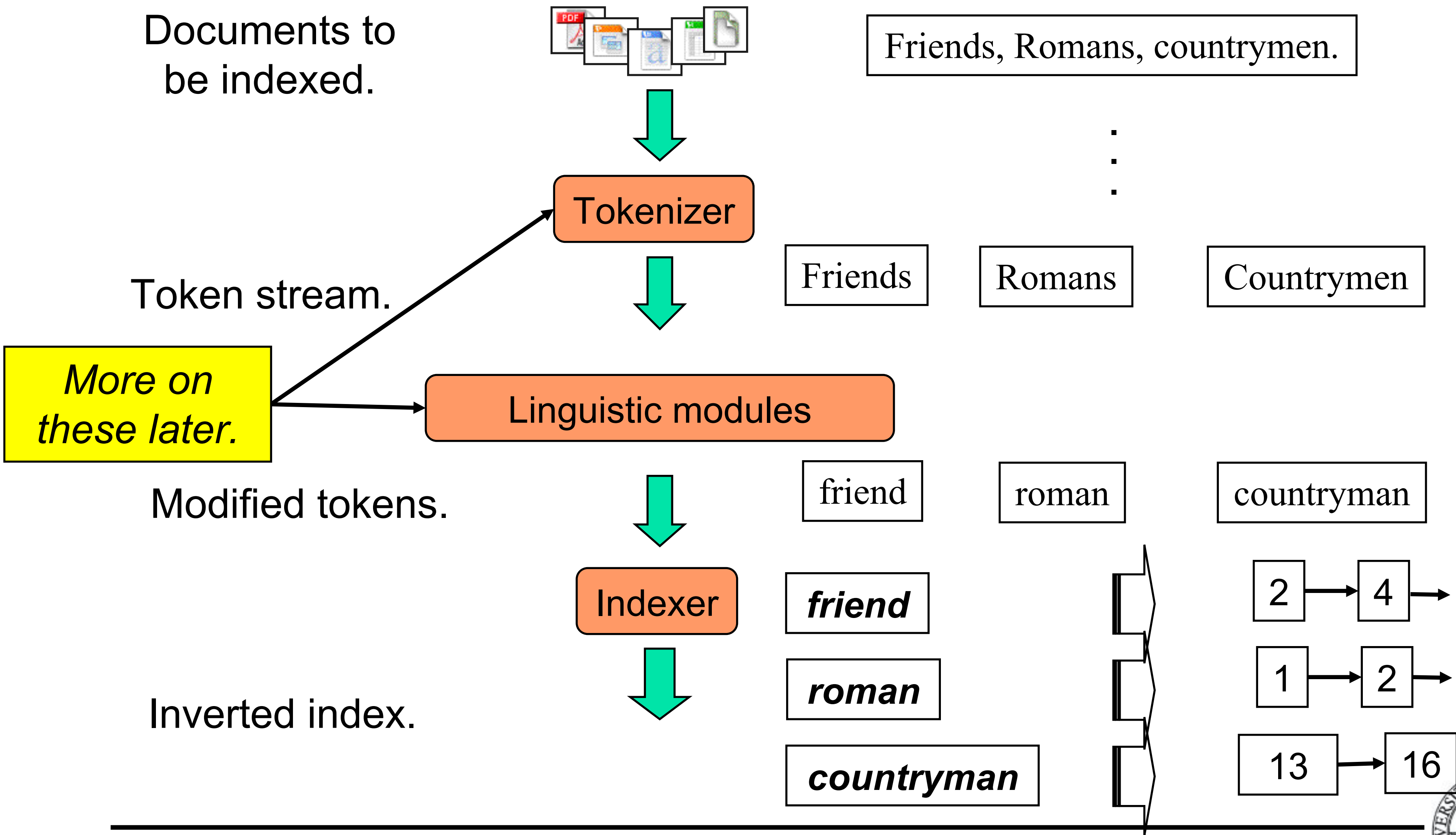
What happens if the word ***Caesar*** is added to document 14?

Inverted index

- Linked lists generally preferred to arrays
  - Dynamic space allocation
  - Insertion of terms into documents easy
  - Space overhead of pointers



Inverted index construction



Indexer steps

- Sequence of (Modified token, Document ID) pairs.

Doc 1

I did enact Julius Caesar I was killed i' the Capitol; Brutus killed me.

Doc 2

So let it be with Caesar. The noble Brutus hath told you Caesar was ambitious

Term	docID
I	1
did	1
enact	1
julius	1
caesar	1
I	1
was	1
killed	1
i'	1
the	1
capitol	1
brutus	1
killed	1
me	1
so	2
let	2
it	2
be	2
with	2
caesar	2
the	2
noble	2
brutus	2
hath	2
told	2
you	2
caesar	2
was	2
ambitious	2

- Sort by terms.

Core indexing step.

Term	docID
I	1
did	1
enact	1
julius	1
caesar	1
I	1
was	1
killed	1
i'	1
the	1
capitol	1
brutus	1
killed	1
me	1
so	2
let	2
it	2
be	2
with	2
caesar	2
the	2
noble	2
brutus	2
hath	2
told	2
you	2
caesar	2
was	2
ambitious	2

Term	docID
ambitious	2
be	2
brutus	1
brutus	2
capitol	1
caesar	1
caesar	2
caesar	2
did	1
enact	1
hath	1
I	1
I	1
i'	1
it	2
julius	1
killed	1
killed	1
let	2
me	1
noble	2
so	2
the	1
the	2
told	2
you	2
was	1
was	2
with	2

Indexer steps: Dictionary & Postings

- Multiple term entries in a single document are merged.
- Split into Dictionary and Postings
- Doc. frequency information is added.

Why frequency?

Will discuss later.

Term	docID
ambitious	2
be	2
brutus	1
brutus	2
capitol	1
caesar	1
caesar	2
caesar	2
did	1
enact	1
hath	1
I	1
I	1
i'	1
it	2
julius	1
killed	1
killed	1
let	2
me	1
noble	2
so	2
the	1
the	2
told	2
you	2
was	1
was	2
with	2

→

term	doc. freq.	→	postings lists
ambitious	1	→	2
be	1	→	2
brutus	2	→	1 → 2
capitol	1	→	1
caesar	2	→	1 → 2
did	1	→	1
enact	1	→	1
hath	1	→	2
i	1	→	1
i'	1	→	1
it	1	→	2
julius	1	→	1
killed	1	→	1
let	1	→	2
me	1	→	1
noble	1	→	2
so	1	→	2
the	2	→	1 → 2
told	1	→	2
you	1	→	2
was	2	→	1 → 2
with	1	→	2

Where do we pay in storage?

term	doc. freq.	→	postings lists
ambitious	1	→	2
be	1	→	2
brutus	2	→	1 → 2
capitol	1	→	1
caesar	2	→	1 → 2
did	1	→	1
enact	1	→	1
hath	1	→	2
i	1	→	1
i'	1	→	1
it	1	→	2
julius	1	→	1
killed	1	→	1
let	1	→	2
me	1	→	1
noble	1	→	2
so	1	→	2
the	2	→	1 → 2
told	1	→	2
you	1	→	2
was	2	→	1 → 2
with	1	→	2

Terms and counts

Lists of docIDs

Pointers

Query processing: AND

Consider processing the query:

Brutus *AND* ***Caesar***

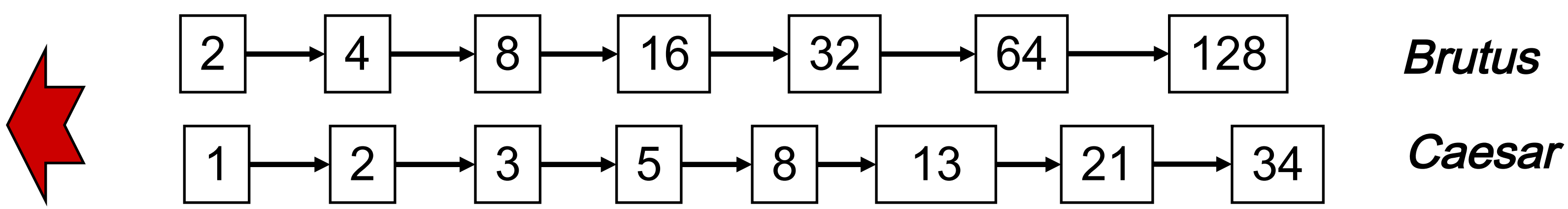
Locate ***Brutus*** in the Dictionary;

Retrieve its postings.

Locate *Caesar* in the Dictionary;

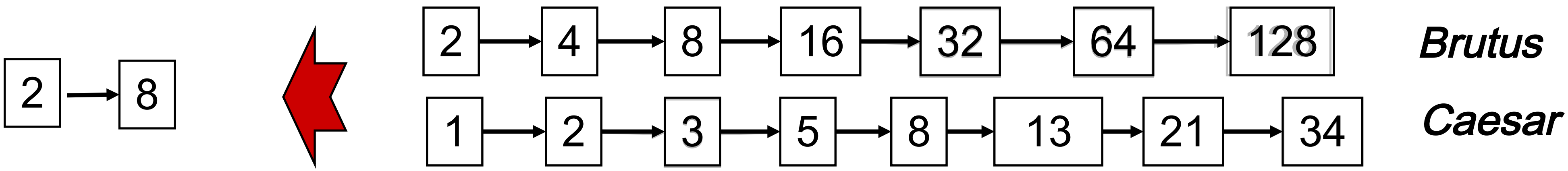
Retrieve its postings.

"Merge" the two postings:



The merge

Walk through the two postings simultaneously, in time linear in the total number of postings entries



If the list lengths are *x* and *y*, the merge takes O(*x*+*y*)

operations.

Crucial: postings sorted by docID.

Boolean queries: Exact match

- The Boolean Retrieval model is being able to ask a query that is a Boolean expression:
  - Boolean Queries are queries using *AND, OR* and *NOT* to join query terms
    - Views each document as a set of words
    - Is precise: document matches condition or not.
- Primary commercial retrieval tool for 3 decades.
- Professional searchers (e.g., lawyers) still like Boolean queries:
  - You know exactly what you're getting.

Evidence accumulation

- 1 vs. 0 occurrence of a search term
  - 2 vs. 1 occurrence
  - 3 vs. 2 occurrences, etc.
  - Usually more seems better
- Need term frequency information in docs

Ranking search results

- Boolean queries give inclusion or exclusion of docs.
- Often we want to rank/group results
  - Need to measure proximity from query to each doc.
  - Need to decide whether docs presented to user are singletons, or a group of docs covering various aspects of the query.

IR vs. databases: Structured vs unstructured data

- Structured data tends to refer to information in "tables"

Employee	Manager	Salary
Smith	Jones	50000
Chang	Smith	60000
Ivy	Smith	50000

Typically allows numerical range and exact match (for text) queries, e.g.,

Salary < 60000 AND Manager = Smith.

Unstructured data

- Typically refers to free-form text
- Allows
  - Keyword queries including operators
  - More sophisticated "concept" queries, e.g.,
    - find all web pages dealing with *drug abuse*
- Classic model for searching text documents

Semi-structured data

- In fact almost no data is "unstructured"
- E.g., this slide has distinctly identified zones such as the *Title* and *Bullets*
- Facilitates "semi-structured" search such as
  - *Title* contains data AND *Bullets* contain search

... to say nothing of linguistic structure

From Binary term-document incidence matrix

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	1	1	0	0	0	1
Brutus	1	1	0	1	0	0
Caesar	1	1	0	1	1	1
Calpurnia	0	1	0	0	0	0
Cleopatra	1	0	0	0	0	0
mercy	1	0	1	1	1	1
worser	1	0	1	1	1	0

Each document is represented by a binary vector $\in \{0,1\}^{|V|}$

To term-document count matrices

- Consider the number of occurrences of a term in a document:
  - Each document is a count vector in $\mathbb{N}^{v}$: a column below

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	157	73	0	0	0	0
Brutus	4	157	0	1	0	0
Caesar	232	227	0	2	1	1
Calpurnia	0	10	0	0	0	0
Cleopatra	57	0	0	0	0	0
mercy	2	0	3	5	5	1
worser	2	0	1	1	1	0

Bag of words model

- Vector representation doesn't consider the ordering of words in a document
- *John is quicker than Mary* and *Mary is quicker than John* have the same vectors
- This is called the bag of words model.
- In a sense, this is a step back: The positional index was able to distinguish these two documents.

Term frequency tf

- The term frequency $\mathrm{tf}_{t,d}$ of term *t* in document *d* is defined as the number of times that *t* occurs in *d*.
- We want to use tf when computing query-document match scores. But how?
- Raw term frequency is not what we want:
  - A document with 10 occurrences of the term is more relevant than a document with 1 occurrence of the term.
  - But not 10 times more relevant.
- Relevance does not increase proportionally with term frequency.

NB: frequency = count in IR

Log-frequency weighting

- The log frequency weight of term t in d is

$$w_{t,d} = \begin{cases} 1 + \log_{10} \mathrm{tf}_{t,d}, & \text{if } \mathrm{tf}_{t,d} > 0 \\ 0, & \text{otherwise} \end{cases}$$

- 0 → 0, 1 → 1, 2 → 1.3, 10 → 2, 1000 → 4, etc.
- Score for a document-query pair: sum over terms *t* in both *q* and *d*:
- score $= \sum_{t \in q \cap d} (1 + \log \mathrm{tf}_{t,d})$
- The score is 0 if none of the query terms is present in the document.

Document frequency

- Rare terms are more informative than frequent terms
  - Recall stop words
- Consider a term in the query that is rare in the collection (e.g., *arachnocentric*)
- A document containing this term is very likely to be relevant to the query *arachnocentric*
- → We want a high weight for rare terms like *arachnocentric*.

idf weight

- df_t is the document frequency of t: the number of documents that contain t
  - df_t is an inverse measure of the informativeness of t
  - $\mathrm{df}_t \leq N$
- We define the idf (inverse document frequency) of t by

$$\mathrm{idf}_t = \log_{10}\,(N/\mathrm{df}_t)$$

  - We use log (N/df_t) instead of N/df_t to "dampen" the effect of idf.

Will turn out the base of the log is immaterial.

tf-idf weighting

- The tf-idf weight of a term is the product of its tf weight and its idf weight.

$$\mathrm{w}_{t,d} = \log(1+\mathrm{tf}_{t,d}) \times \log_{10}(N/\mathrm{df}_t)$$

- Best known weighting scheme in information retrieval
  - Note: the "-" in tf-idf is a hyphen, not a minus sign!
  - Alternative names: tf.idf, tf x idf
- Increases with the number of occurrences within a document
- Increases with the rarity of the term in the collection

Score for a document given a query

$$\mathrm{Score}(q,d) = \sum_{t \in q \cap d} \mathrm{tf} \times \mathrm{idf}_{t,d}$$

- There are many variants
  - How "tf" is computed (with/without logs)
  - Whether the terms in the query are also weighted
  - …

Binary → count → weight matrix

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	5.25	3.18	0	0	0	0.35
Brutus	1.21	6.1	0	1	0	0
Caesar	8.59	2.54	0	1.51	0.25	0
Calpurnia	0	1.54	0	0	0	0
Cleopatra	2.85	0	0	0	0	0
mercy	1.51	0	1.9	0.12	5.25	0.88
worser	1.37	0	0.11	4.15	0.25	1.95

Each document is now represented by a real-valued vector of tf-idf weights $\in R|V|$

Documents as vectors

- So we have a |V|-dimensional vector space
- Terms are axes of the space
- Documents are points or vectors in this space
- Very high-dimensional: tens of millions of dimensions when you apply this to a web search engine
- These are very sparse vectors - most entries are zero.

Queries as vectors

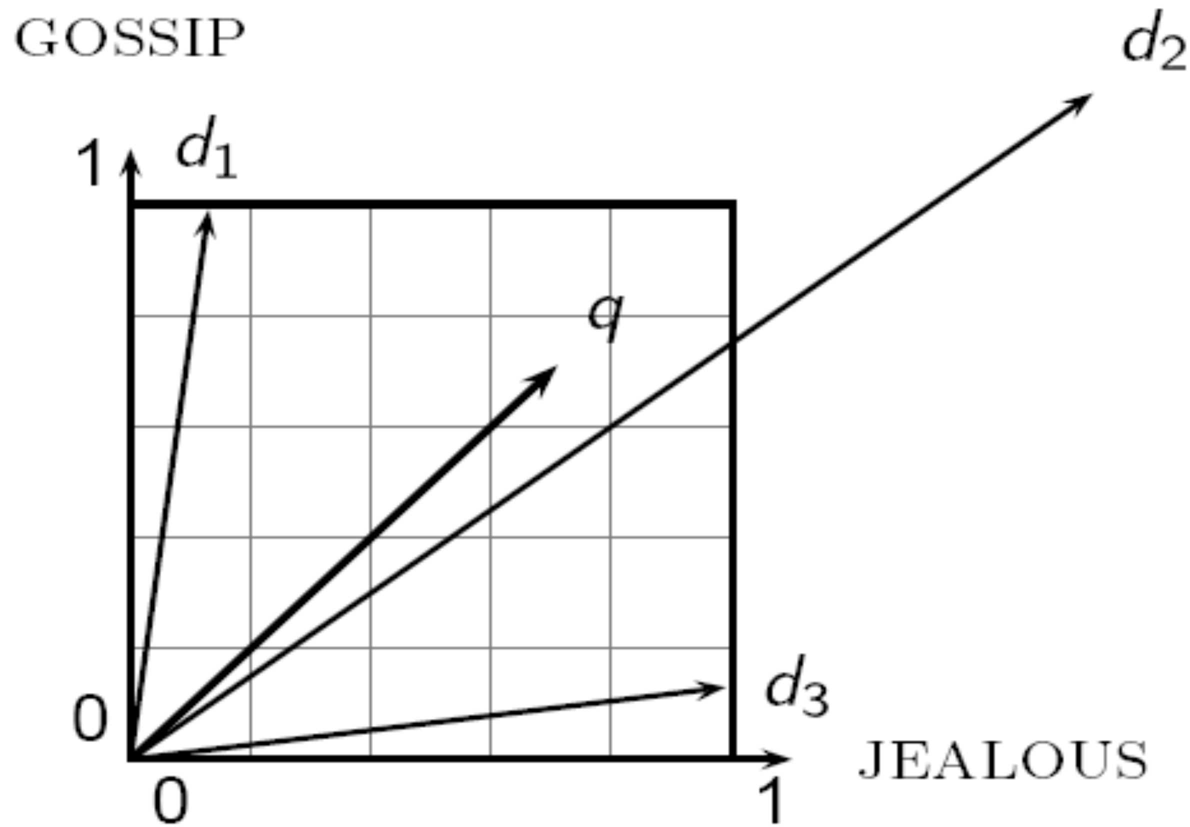
- Key idea 1: Do the same for queries: represent them as vectors in the space
- Key idea 2: Rank documents according to their proximity to the query in this space
  - proximity = similarity of vectors
  - proximity ≈ inverse of distance
  - rank more relevant documents higher than less relevant documents

Formalizing vector space proximity

- First cut: distance between two points
  - (= distance between the end points of the two vectors)
- Euclidean distance?
- Euclidean distance is a bad idea . . .
- . . . because Euclidean distance is large for vectors of different lengths.

Why distance is a bad idea

The Euclidean distance between $\vec{q}$ and $\vec{d_2}$ is large even though the distribution of terms in the query $\vec{q}$ and the distribution of terms in the document $\vec{d_2}$ are very similar.



Use angle instead of distance

- Thought experiment: take a document *d* and append it to itself. Call this document *d*′.
- "Semantically" d and d′ have the same content
- The Euclidean distance between the two documents can be quite large
- The angle between the two documents is 0, corresponding to maximal similarity.

- Key idea: Rank documents according to angle with query.

From angles to cosines

- The following two notions are equivalent.
  - Rank documents in decreasing order of the angle between query and document
  - Rank documents in increasing order of cosine (query,document)
- Cosine is a monotonically decreasing function for the interval [0^{o}, 180^{o}]

Length normalization

- A vector can be (length-) normalized by dividing each of its components by its length – for this we use the L_2 norm:

$$\|\vec{x}\|_2 = \sqrt{\sum_i x_i^2}$$

- Dividing a vector by its L_2 norm makes it a unit (length) vector (on surface of unit hypersphere)
- Effect on the two documents d and d′ (d appended to itself) from earlier slide: they have identical vectors after length-normalization.
  - Long and short documents now have comparable weights

cosine(query,document)

Dot product

$$\cos(\vec{q},\vec{d}) = \frac{\vec{q} \bullet \vec{d}}{|\vec{q}||\vec{d}|} = \frac{\vec{q}}{|\vec{q}|} \bullet \frac{\vec{d}}{|\vec{d}|} = \frac{\sum_{i=1}^{|V|} q_i d_i}{\sqrt{\sum_{i=1}^{|V|} q_i^2}\sqrt{\sum_{i=1}^{|V|} d_i^2}}$$

qi is the tf-idf weight of term *i* in the query
di is the tf-idf weight of term *i* in the document

$\cos(\vec{q},\vec{d})$ is the cosine similarity of $\vec{q}$ and $\vec{d}$ … or, equivalently, the cosine of the angle between $\vec{q}$ and $\vec{d}$.

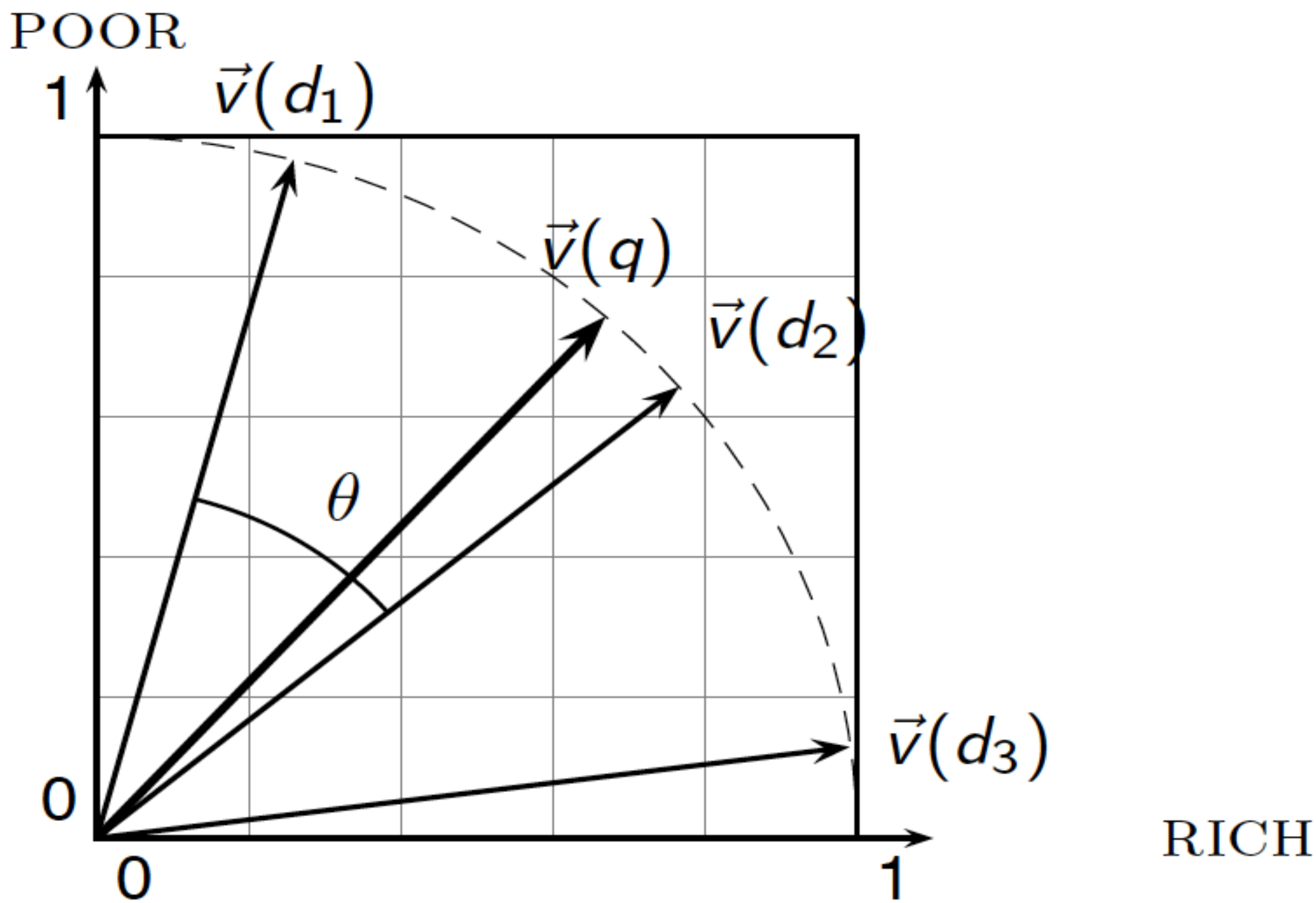
Cosine for length-normalized vectors

- For length-normalized vectors, cosine similarity is simply the dot product (or scalar product):

$$\cos(\vec{q},\vec{d}) = \vec{q} \bullet \vec{d} = \sum_{i=1}^{|V|} q_i d_i$$

for q, d length-normalized.

Cosine similarity illustrated



Performance Evaluation

Measures for a search engine

- We can quantify speed/size
- Quality of the retrieved documents
- Relevance measurement requires 3 elements:
  1. A benchmark document collection
  2. A benchmark suite of queries
  3. A usually binary assessment of either Relevant or Non relevant for each query and each document
     - Some work on more-than-binary, but not the standard

Evaluating an IR system

- Note: the **information need** is translated into a **query**
- Relevance is assessed relative to the **information need** *not* the **query**
- E.g., Information need: *I'm looking for information on whether drinking red wine is more effective at reducing your risk of heart attacks than white wine.*
- Query: ***wine red white heart attack effective***
- Evaluate whether the doc addresses the information need, not whether it has these words

Standard relevance benchmarks

- TREC - National Institute of Standards and Technology (NIST) has run a large IR test bed for many years
- Reuters and other benchmark doc collections used
- "Retrieval tasks" specified
  - sometimes as queries
- Human experts mark, for each query and for each doc, Relevant or Nonrelevant
  - or at least for subset of docs that some system returned for that query

Unranked retrieval evaluation: Precision and Recall

- **Precision**: fraction of retrieved docs that are relevant
  = P(relevant|retrieved)
- **Recall**: fraction of relevant docs that are retrieved
  = P(retrieved|relevant)

	Relevant	Nonrelevant
Retrieved	tp	fp
Not Retrieved	fn	tn

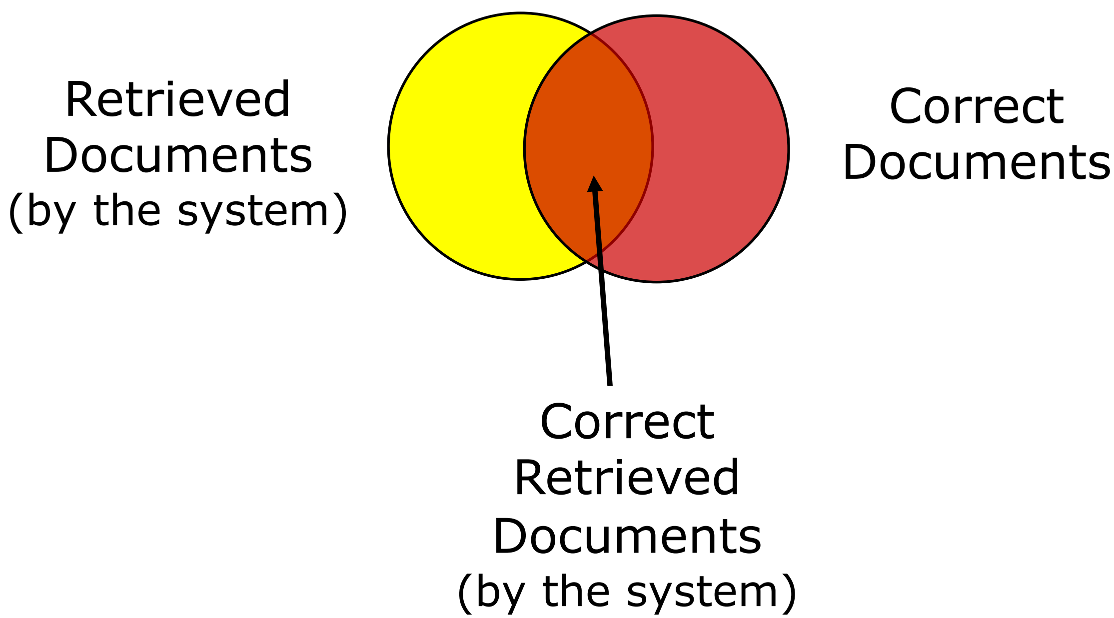
- Precision P = tp/(tp + fp)
- Recall R = tp/(tp + fn)

Should we instead use the accuracy measure for evaluation?

- Given a query, an engine classifies each doc as "Relevant" or "Nonrelevant"
- The **accuracy** of an engine: the fraction of these classifications that are correct
  - (tp + tn) / (tp + fp + fn + tn)
- **Accuracy** is a evaluation measure in often used in machine learning classification work
- Why is this not a very useful evaluation measure in IR?

Performance Measurements

- Given a set of document *T*
- Precision = # Correct Retrieved Document / # Retrieved Documents
- Recall = # Correct Retrieved Document/ # Correct Documents



Why not just use accuracy?

- How to build a 99.9999% accurate search engine on a low budget….

snoogle.com

Search for:

0 matching results found.

- People doing information retrieval *want to find something* and have a certain tolerance for junk.

Precision/Recall trade-off

- You can get high recall (but low precision) by retrieving all docs for all queries!
- Recall is a non-decreasing function of the number of docs retrieved

- In a good system, precision decreases as either the number of docs retrieved or recall increases
  - This is not a theorem, but a result with strong empirical confirmation

A combined measure: *F*

- Combined measure that assesses precision/recall tradeoff is **F measure** (weighted harmonic mean):

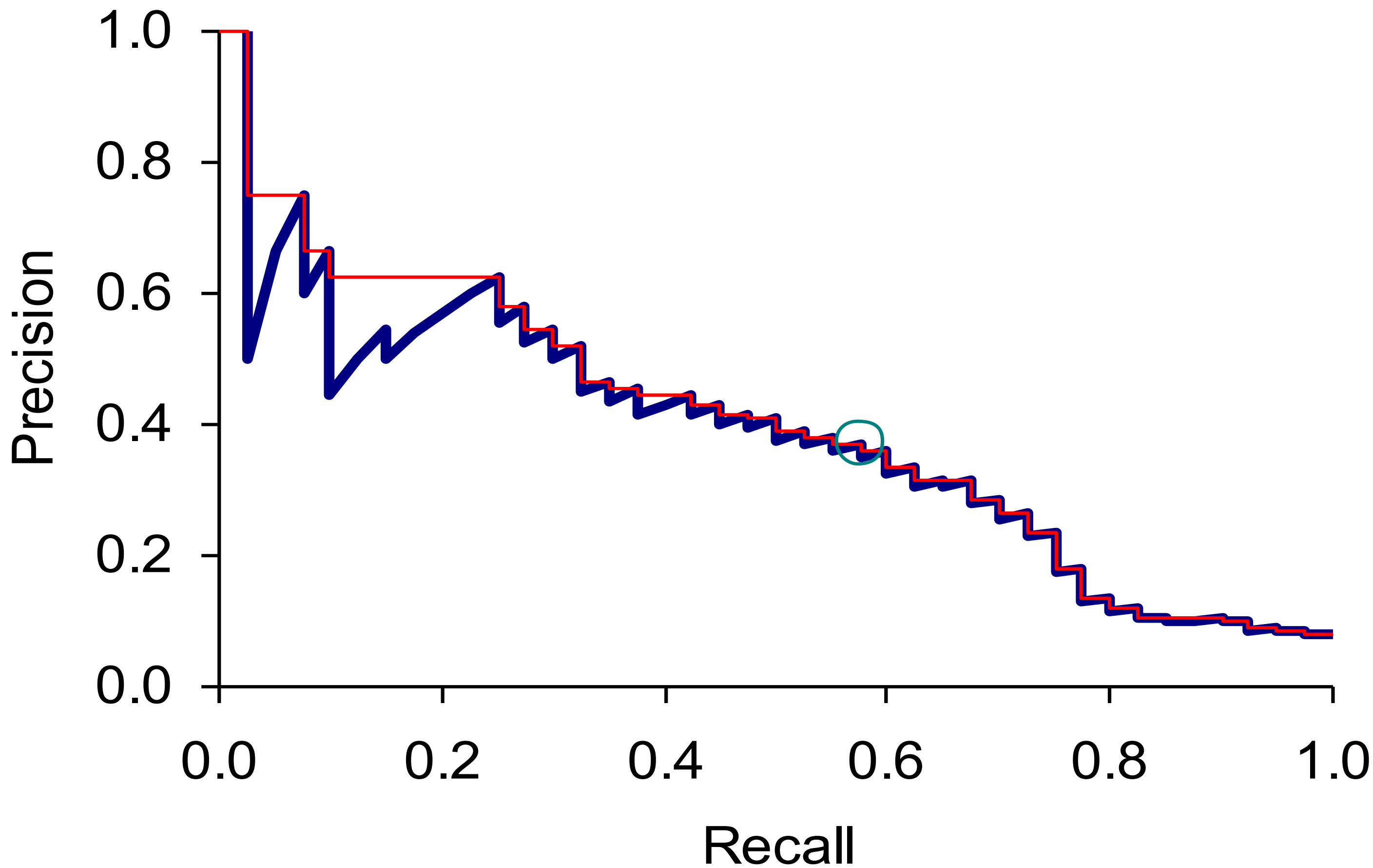
$$F = \frac{1}{\alpha\frac{1}{P} + (1-\alpha)\frac{1}{R}} = \frac{(\beta^2+1)PR}{\beta^2 P + R}$$

- People usually use balanced F_1 measure
  - i.e., with $\beta = 1$ or $\alpha = ½$
- Harmonic mean is a conservative average
  - See CJ van Rijsbergen, *Information Retrieval*

Evaluating ranked results

- Evaluation of ranked results:
  - The system can return any number of results
  - By taking various numbers of the top returned documents (levels of recall), the evaluator can produce a *precision-recall curve*

A precision-recall curve



Averaging over queries

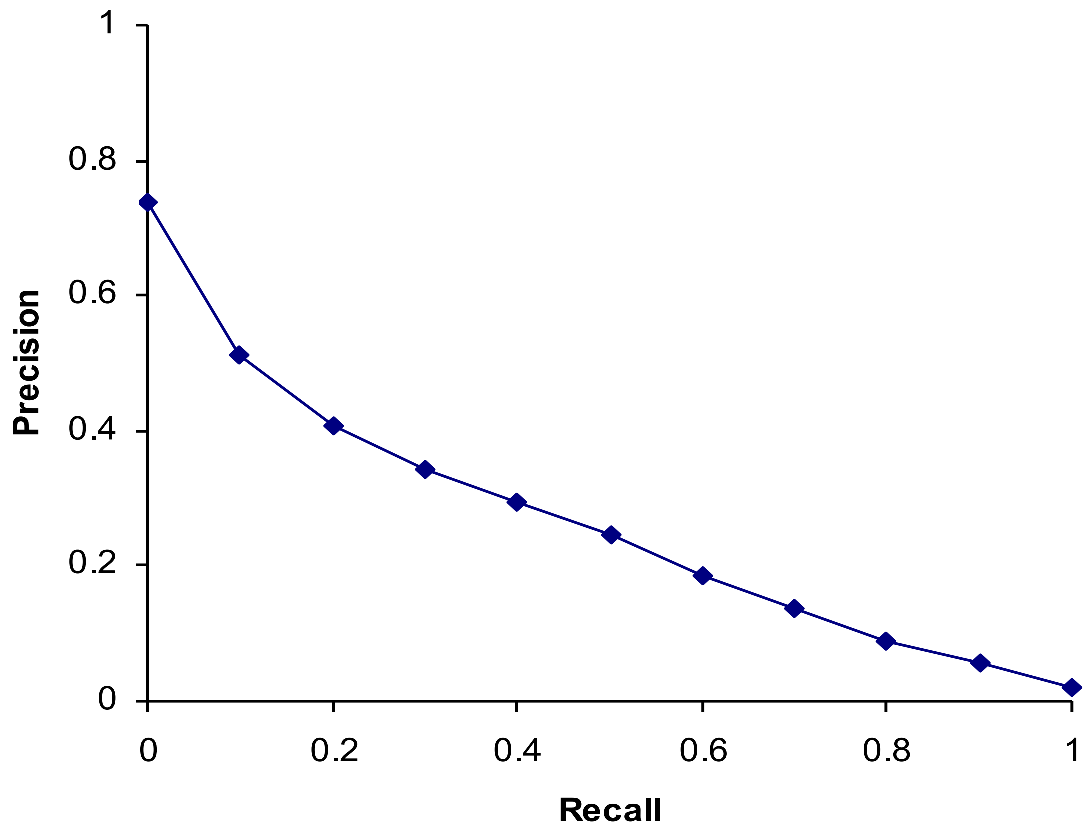
- A precision-recall graph for one query isn't a very sensible thing to look at
- You need to average performance over a whole bunch of queries.
- But there's a technical issue:
  - Precision-recall calculations place some points on the graph
  - How do you determine a value (interpolate) between the points?

Evaluation

- Graphs are good, but people want summary measures!
  - Precision at fixed retrieval level
    - Precision-at-*k*: Precision of top *k* results
    - Perhaps appropriate for most of web search: all people want are good matches on the first one or two results pages
    - But: averages badly and has an arbitrary parameter of *k*
  - 11-point interpolated average precision
    - The standard measure in the early TREC competitions: you take the precision at 11 levels of recall varying from 0 to 1 by tenths of the documents, using interpolation (the value for 0 is always interpolated!), and average them
    - Evaluates performance at all recall levels

Typical (good) 11 point precisions

- SabIR/Cornell 8A1 11pt precision from TREC 8 (1999)



Yet more evaluation measures…

- Mean average precision (MAP)
  - Average of the precision value obtained for the top *k* documents, each time a relevant doc is retrieved
  - Avoids interpolation, use of fixed recall levels
  - MAP for query collection is arithmetic ave.
    - Macro-averaging: each query counts equally
- R-precision
  - If we have a known (though perhaps incomplete) set of relevant documents of size *Rel,* then calculate precision of the top *Rel* docs returned
  - Perfect system could score 1.0.

TREC

- TREC Ad Hoc task from first 8 TRECs is standard IR task
  - 50 detailed information needs a year
  - Human evaluation of pooled results returned
  - More recently other related things: Web track, HARD
- A TREC query (TREC 5)

  <top>

  <num> Number: 225

  <desc> Description:

  What is the main function of the Federal Emergency Management Agency (FEMA) and the funding level provided to meet emergencies? Also, what resources are available to FEMA such as people, equipment, facilities?

  </top>

Standard relevance benchmarks: Others

- GOV2
  - Another TREC/NIST collection
  - 25 million web pages
  - Largest collection that is easily available
  - But still 3 orders of magnitude smaller than what Google/Yahoo/MSN index
- NTCIR
  - East Asian language and cross-language information retrieval
- Cross Language Evaluation Forum (CLEF)
  - This evaluation series has concentrated on European languages and cross-language information retrieval.
- Many others

Text Categorization

Text Classification Problem

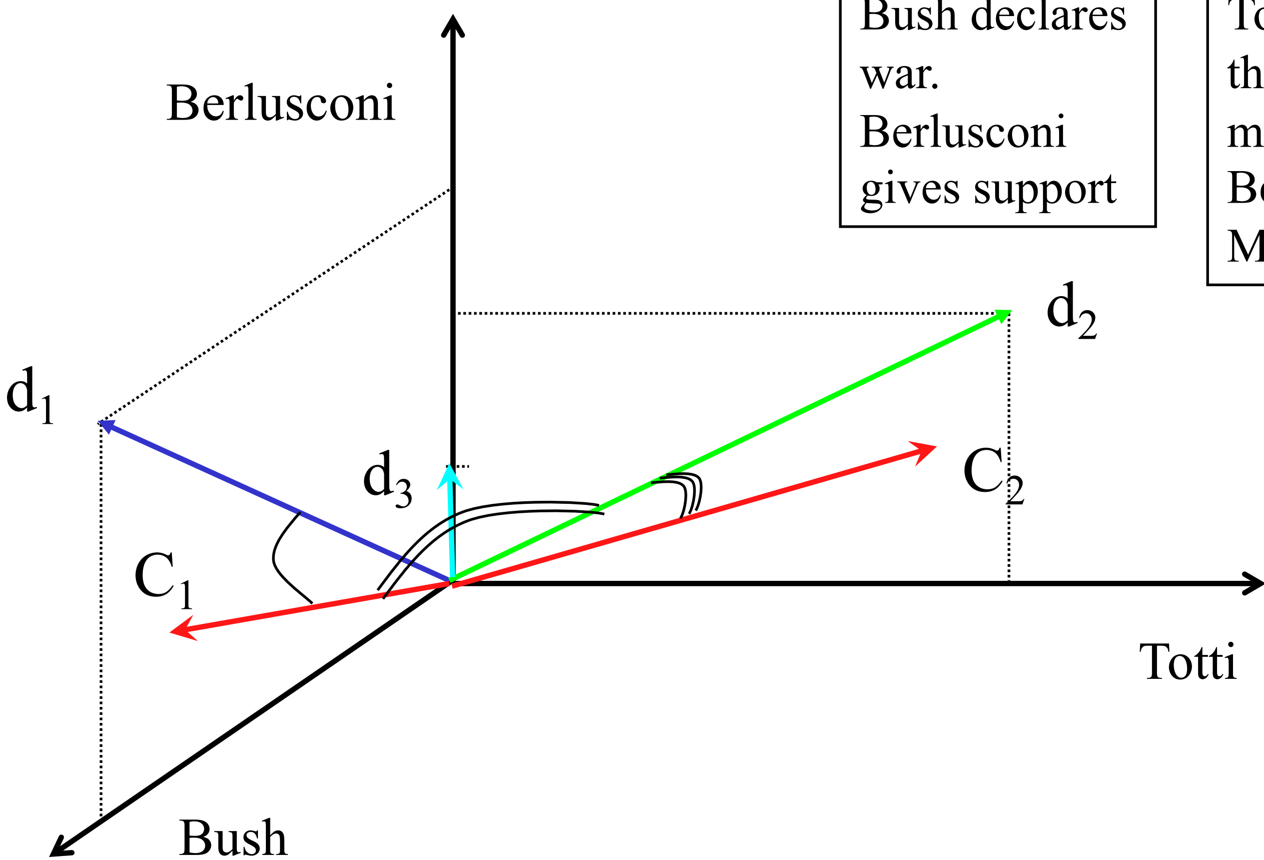
- Given:
  - a set of target categories:
  - the set T of documents, $C = \left\{ C^1, .., C^n \right\}$

  define

  $$f : T \rightarrow 2^C$$

- VSM (Salton89')
  - Features are dimensions of a Vector Space.
  - Documents and Categories are vectors of feature weights.
  - d is assigned to C^i if $\vec{d} \cdot \vec{C}^i > th$

The Vector Space Model



d_1: Politic

Bush declares war. Berlusconi gives support

d_2: Sport

Totti scored in the yesterday match against Berlusconi's Milan

d_3:Economic

Berlusconi acquires Inzaghi before elections

C_1 : Politics Category

C_2 : Sport Category

Automated Text Categorization

- A corpus of pre-categorized documents
- Split document in two parts:
  - Training-set
  - Test-set
- Apply a supervised machine learning model to the training-set
  - Positive examples
  - Negative examples
- Measure the performances on the test-set
  - e.g., Precision and Recall

Feature Vectors

- Each example is associated with a vector of *n* feature types (e.g. unique words in TC)

$$\vec{x} = (0,\ ..,1,..,0,..,0,\ ..,1,..,0,..,0,\ ..,1,..,0,..,0,\ ..,1,..,0,..,\ 1)$$

acquisition buy market sell stocks

- The dot product $\vec{x} \cdot \vec{z}$ counts the number of features in common

- This provides a sort of *similarity*

Text Categorization phases

- Corpus pre-processing (e.g. tokenization, stemming)
- Feature Selection (optionally)
  - Document Frequency, Information Gain, χ_2 , mutual information,...
- Feature weighting
  - for documents and profiles
- Similarity measure
  - between document and profile (e.g. scalar product)
- Statistical Inference
  - threshold application
- Performance Evaluation
  - Accuracy, Precision/Recall, BEP, f-measure,..

Feature Selection

- Some words, i.e. features, may be irrelevant
- For example, “function words” as: “the”, “on”,”those”…
- Two benefits:
  - efficiency
  - Sometime the accuracy
- Sort features by relevance and select the *m*-best

Statistical Quantity to sort feature

- Based on corpus counts of the pair

- A is the number of documents in which both f and c occur, i.e. (f, c);
- B is the number of documents in which only f occurs, i.e. $(f, \bar{c})$;
- C is the number of documents in which only c occurs, i.e. $(\bar{f}, c)$;
- D is the number of documents in which neither f nor c occur, i.e. $(\bar{f}, \bar{c})$;
- N is the total number of documents, i.e. $A + B + C + D$.

Statistical Selectors

- Chi-square, Pointwise MI and MI

$$\chi^2(f,c) = \frac{N \times (AD - CB)^2}{(A+C)(B+D)(A+B)(C+D)}$$

$$PMI(f,c) = log\frac{P(f,c)}{P(f) \times P(c)}$$

$$\begin{aligned} MI(f) \\ (f,C) &= -\sum_{c\in\mathcal{C}} P(c)log(P(c)) + P(f)\sum_{c\in\mathcal{C}} P(c|f)log(P(c|f)) \\ &+P(\bar{f})\sum_{c\in\mathcal{C}} P(c|\bar{f})log(P(c|\bar{f})) \end{aligned}$$

Profile Weighting: the Rocchio's formula

- ω_f^d, the weight of *f* in *d*
  - Several weighting schemes (e.g. TF * IDF, Salton 91')

- $\vec{C}_f^i$, the profile weights of *f* in C_i:

$$\vec{C}_f^i = \max\left\{ 0, \frac{\beta}{|T_i|} \sum_{d \in T_i} \omega_f^d - \frac{\gamma}{|\overline{T}_i|} \sum_{d \in \overline{T}_i} \omega_f^d \right\}$$

- T_i, the training documents in C^i

Similarity estimation

- Given the document and the category representation

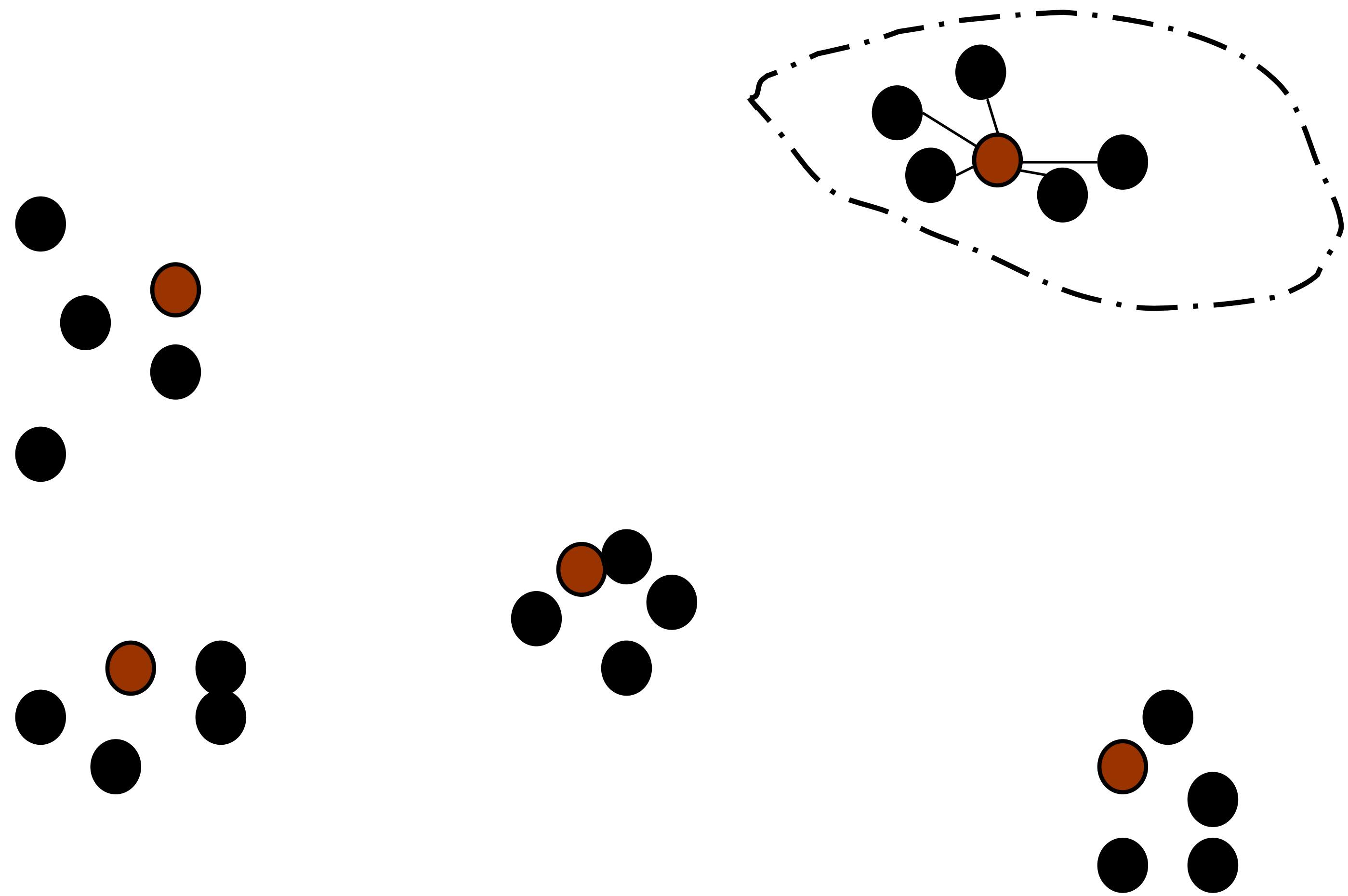
$$\vec{d} = \left\langle \omega^d_{f_1}, \ldots, \omega^d_{f_n} \right\rangle, \quad \vec{C}_i = \left\langle \Omega^i_{f_1}, \ldots, \Omega^i_{f_n} \right\rangle$$

- It can be defined the following similarity function (cosine measure

$$s_{d,i} = \cos(\vec{d}, \vec{C}_i) = \frac{\vec{d} \cdot \vec{C}^i}{\|\vec{d}\| \times \|\vec{C}_i\|} = \frac{\sum_f \omega^d_f \times \Omega^i_f}{\|\vec{d}\| \times \|\vec{C}_i\|}$$

- *d* is assigned to C^i if $\vec{d} \cdot \vec{C}^i > \sigma$

Clustering



Experiments

- Reuters Collection 21578 Apté split (Apté94)
  - 90 classes (12,902 docs)
  - A fixed splitting between training and test set
  - 9603 vs 3299 documents
- Tokens
  - about 30,000 different
- Other different versions have been used but …

  most of TC results relate to the 21578 Apté
  - [Joachims 1998], [Lam and Ho 1998], [Dumais et al. 1998], [Li Yamanishi 1999], [Weiss et al. 1999],

    [Cohen and Singer 1999]…

A Reuters document- Acquisition Category

CRA SOLD FORREST GOLD FOR 76 MLN DLRS - WHIM CREEK

SYDNEY, April 8 - <Whim Creek Consolidated NL> said the consortium it is leading will pay 76.55 mln dlrs for the acquisition of CRA Ltd's <CRAA.S> <Forrest Gold Pty Ltd> unit, reported yesterday.

CRA and Whim Creek did not disclose the price yesterday.

Whim Creek will hold 44 pct of the consortium, while <Austwhim Resources NL> will hold 27 pct and <Croesus Mining NL> 29 pct, it said in a statement.

As reported, Forrest Gold owns two mines in Western Australia producing a combined 37,000 ounces of gold a year. It also owns an undeveloped gold project.

A Reuters document- Crude-Oil Category

FTC URGES VETO OF GEORGIA GASOLINE STATION BILL

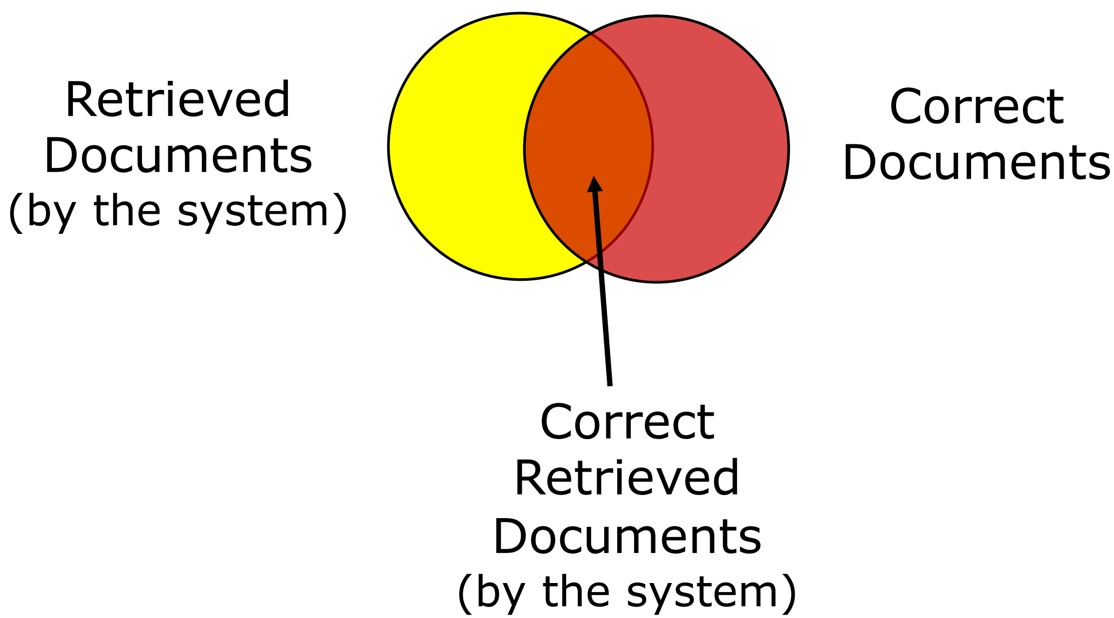
WASHINGTON, March 20 - The Federal Trade Commission said its staff has urged the governor of Georgia to veto a bill that would prohibit petroleum refiners from owning and operating retail gasoline stations.

The proposed legislation is aimed at preventing large oil refiners and marketers from using predatory or monopolistic practices against franchised dealers.

But the FTC said fears of refiner-owned stations as part of a scheme of predatory or monopolistic practices are unfounded. It called the bill anticompetitive and warned that it would force higher gasoline prices for Georgia motorists.

Performance Measurements

- Given a set of document *T*
- Precision = # Correct Retrieved Document / # Retrieved Documents
- Recall = # Correct Retrieved Document/ # Correct Documents



Precision and Recall of C_i

- a, corrects
- b, mistakes

The *Precision* and *Recall* are defined by the above counts:

$$Precision_i = \frac{a_i}{a_i + b_i}$$

$$Recall_i = \frac{a_i}{a_i + c_i}$$

Performance Measurements (cont'd)

- Breakeven Point
  - Find thresholds for which
    Recall = Precision
  - Interpolation
- f-measure
  - Harmonic mean between precision and recall
- Global performance on more than two categories
  - Micro-average
    - The counts refer to classifiers
  - Macro-average (average measures over all categories)

F-measure e MicroAverages

$$F_1 = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

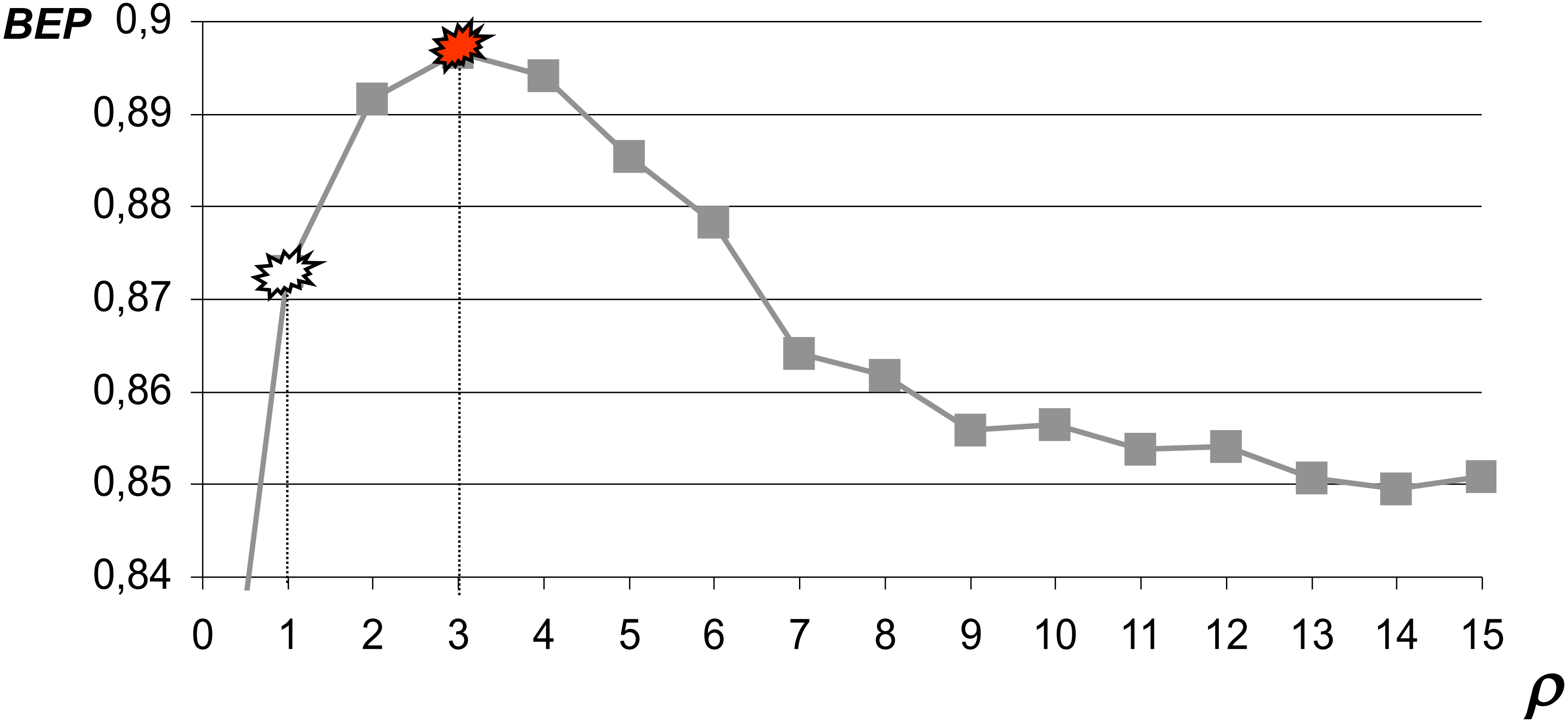
$$\mu Precision = \frac{\sum_{i=1}^{n} a_i}{\sum_{i=1}^{n} a_i + b_i}$$

$$\mu Recall = \frac{\sum_{i=1}^{n} a_i}{\sum_{i=1}^{n} a_i + c_i}$$

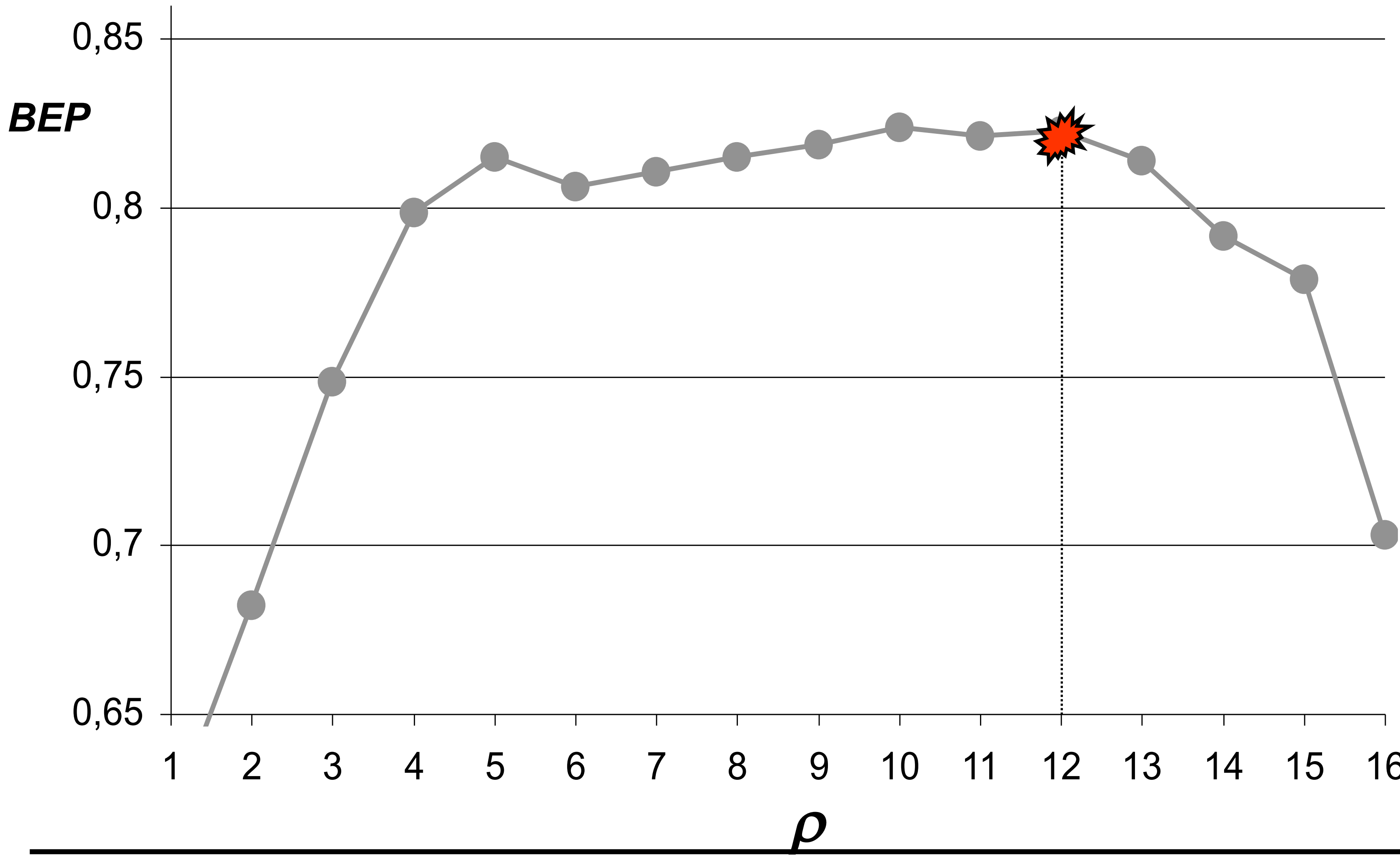
$$\mu BEP = \frac{\mu Precision + \mu Recall}{2}$$

$$\mu f_1 = \frac{2 \times \mu Precision \times \mu Recall}{\mu Precision + \mu Recall}$$

The Impact of ρ parameter on Acquisition category



The impact of ρ parameter on Trade category



N-fold cross validation

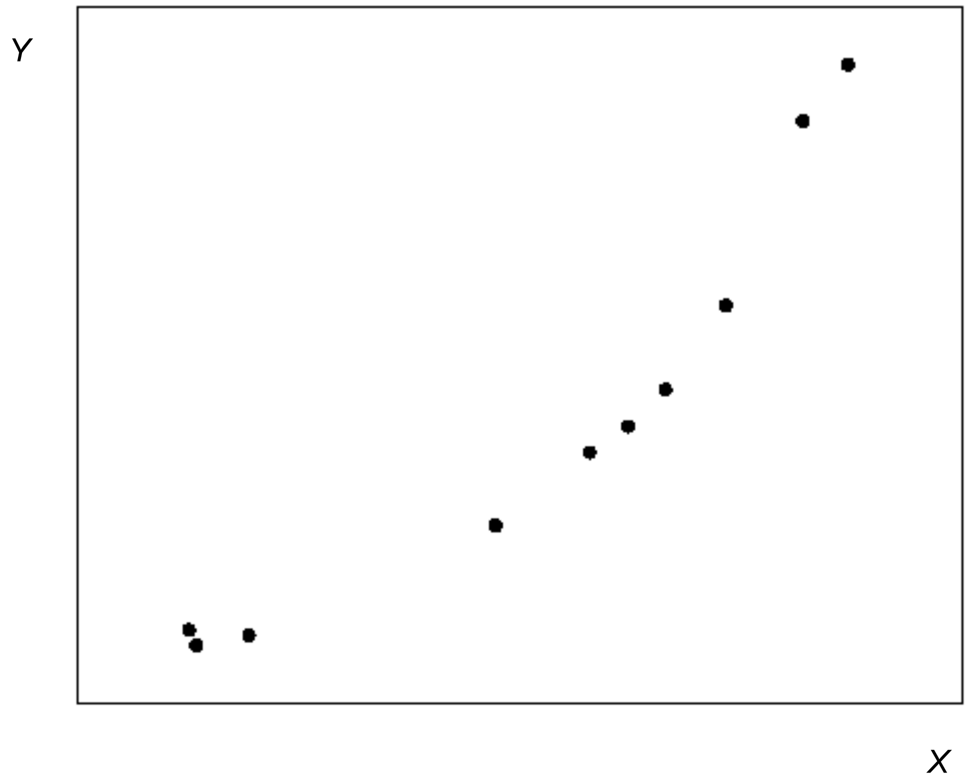
- Divide training set in *n* parts
  - One is used for testing
  - *n-1* for training
- This can be repeated *n* times for *n* distinct test sets
- Average and Std. Dev. are the final performance index

Introduction to Machine Learning

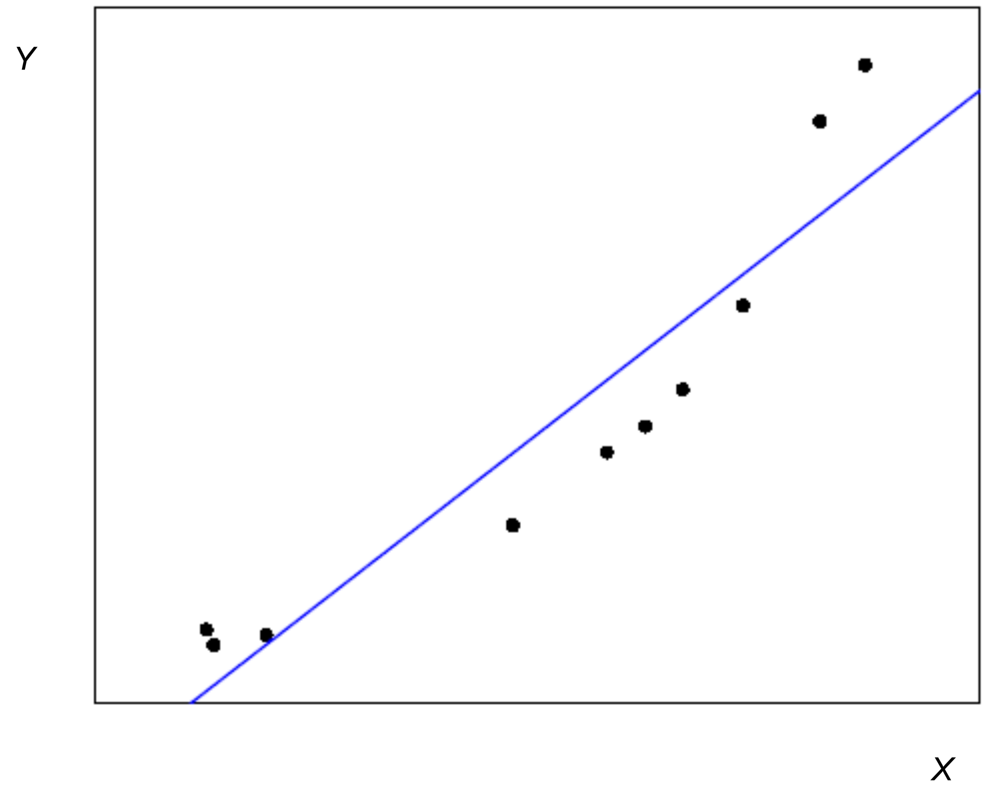
What is Statistical Learning?

- Statistical Methods – Algorithms that learn relations in the data from examples
- Simple relations are expressed by pairs of variables: $\langle x_1, y_1 \rangle$, $\langle x_2, y_2 \rangle, \ldots,$ $\langle x_n, y_n \rangle$
- Learning *f* such that evaluate y^* given a new value x^*, i.e. $\langle x^*, f(x^*) \rangle = \langle x^*, y^* \rangle$

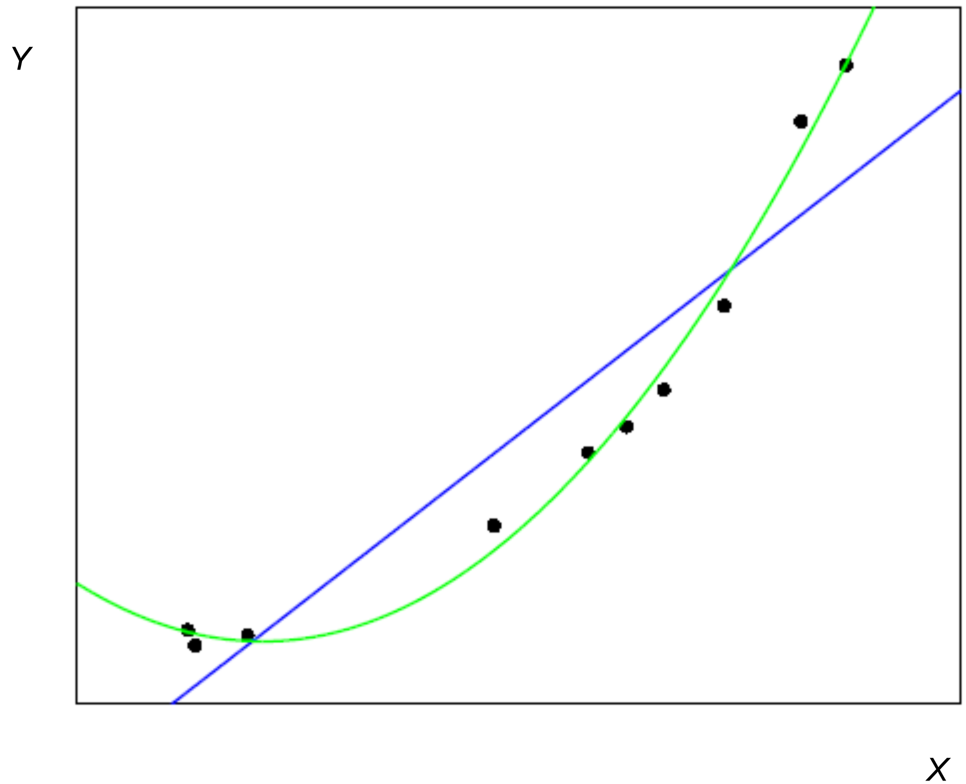
You have already tackled the learning problem



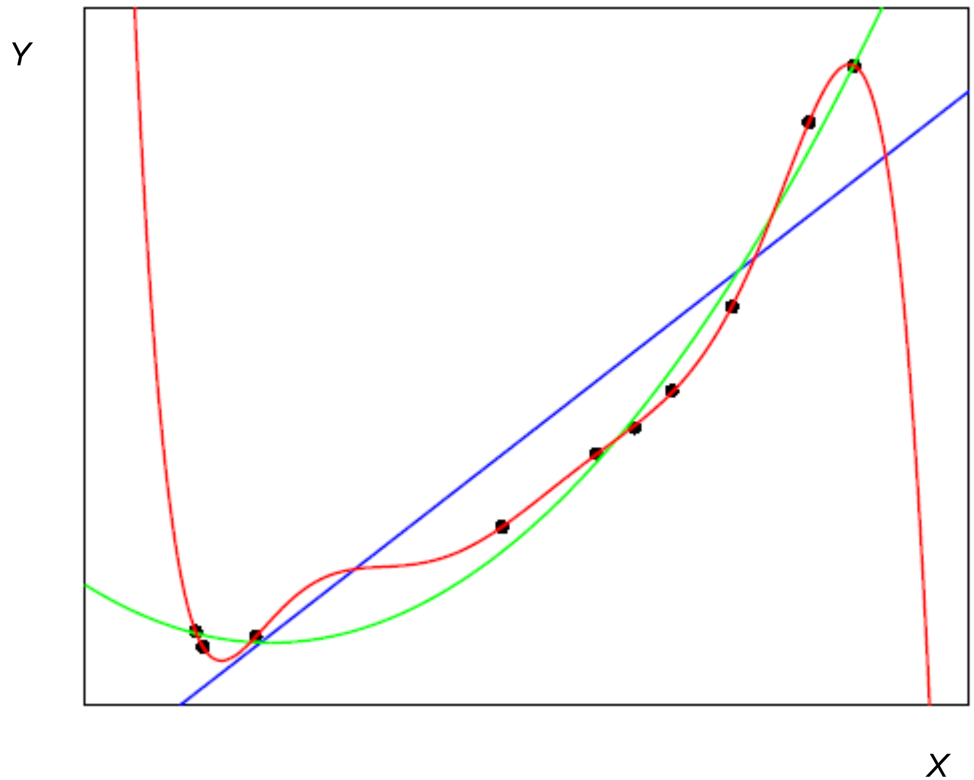
Linear Regression



Degree 2



Degree



Machine Learning Problems

- Overfitting
- How dealing with millions of variables instead of only two?
- How dealing with real world objects instead of real values?

Linear Classifiers

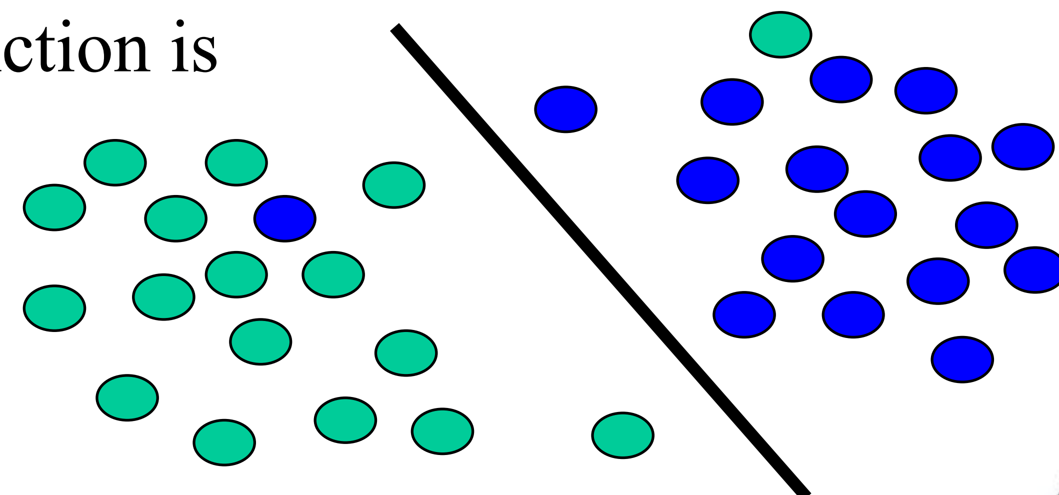
Linear Classifier (1)

- The equation of a hyperplane is

$$f(\vec{x}) = \vec{x} \cdot \vec{w} + b = 0, \quad \vec{x}, \vec{w} \in \Re^n, b \in \Re$$

- $\vec{x}$ is the vector representing the classifying example
- $\vec{w}$ is the gradient to the hyperplane
- The classification function is

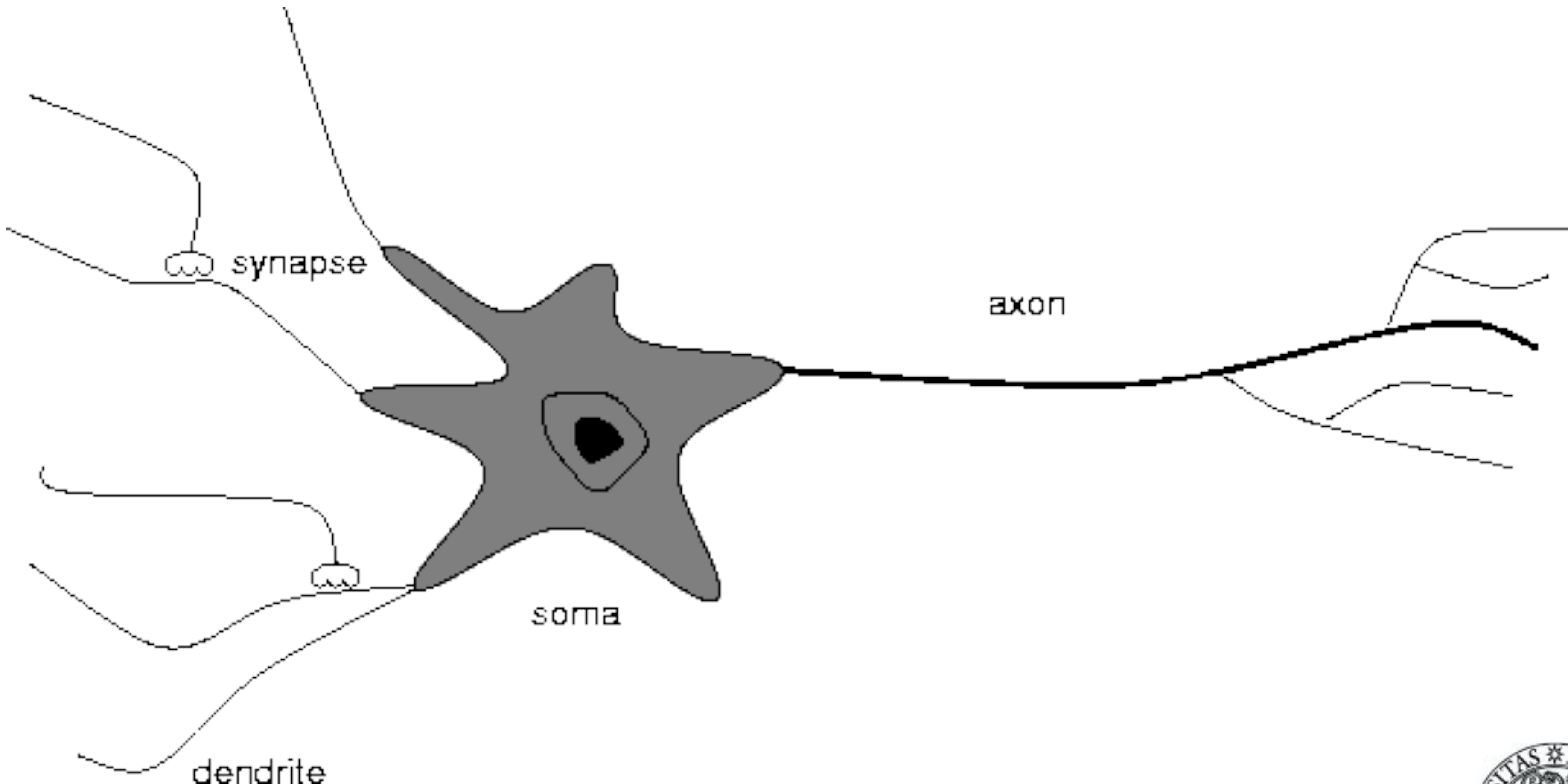
$$h(x) = \text{sign}(f(x))$$



Linear classifiers (2)

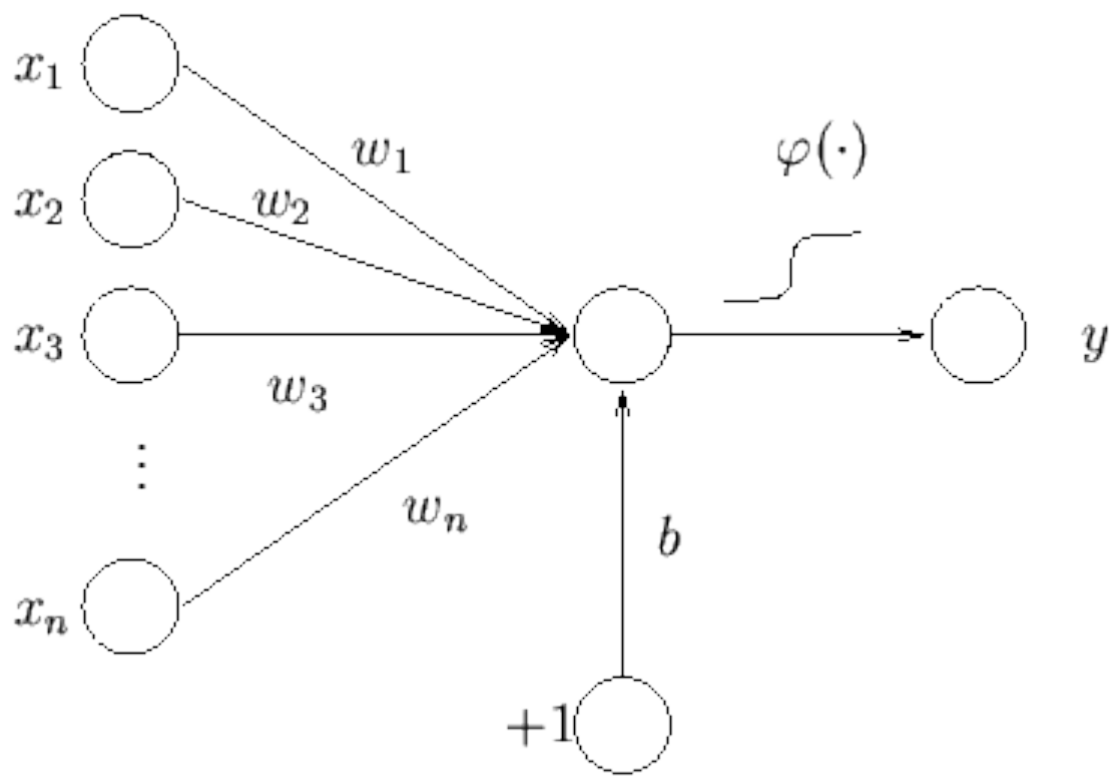
- Linear Functions are the simplest ones from an analytical point of view.
- The basic idea is to select a hypothesis with null error on the training-set.
- To learn a linear function a simple neural network of only one neuron is enough (Perceptron)

An animal neuron



The Perceptron

$$\varphi(\vec{x}) = \operatorname{sgn}\left(\sum_{i=1..n} w_i \times x_i + b \right)$$



Useful Concepts

- ***Functional Margin*** of an example with respect to a hyperplane: $\gamma_i = y_i(\vec{w} \cdot \vec{x}_i + b)$
- ***The distribution of functional margins*** of a hyperplane with respect to a training set *S* is the distribution of the margins of the examples in *S wrt* the hyperplane $(\vec{w}, b)$.
- ***The functional margin of a hyperplane*** is the minimum margin of the distribution

Notations (con'td)

- If we normalize the hyperplane equation, i.e.

$$\left(\frac{\vec{w}}{\|\vec{w}\|}, \frac{b}{\|\vec{w}\|} \right)$$

, we obtain the ***geometric margin***

- The ***geometric margin*** measure the Euclidean distance between the target point and the hyperplane.
- ***The training set Margin*** is the maximum geometric (functional) margin among all hyperplanes which separates the examples in S.
- The hyperplane associated with the above quantity is called ***maximal margin hyperplane***

Basic Concepts

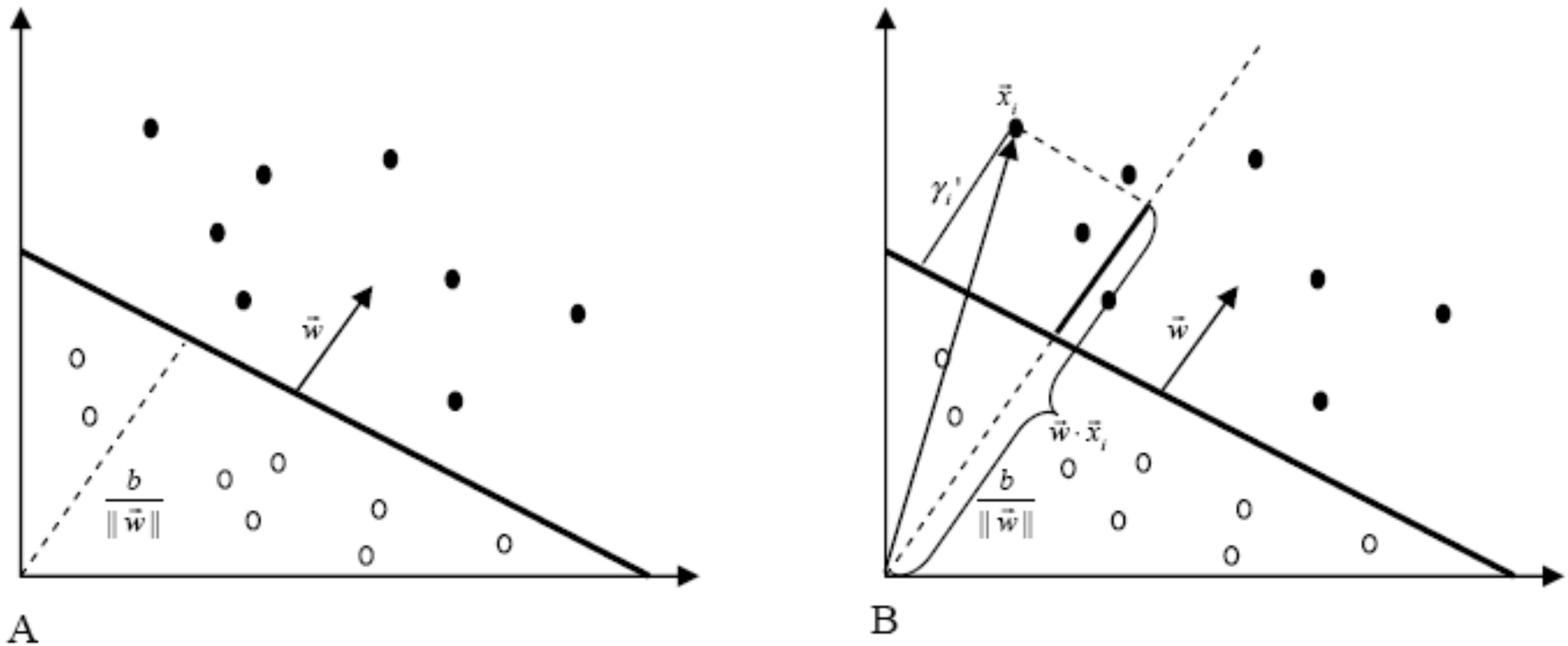
- From $\cos(\vec{x}, \vec{w}) = \dfrac{\vec{x} \cdot \vec{w}}{\|\vec{x}\| \cdot \|\vec{w}\|}$

- It follows that

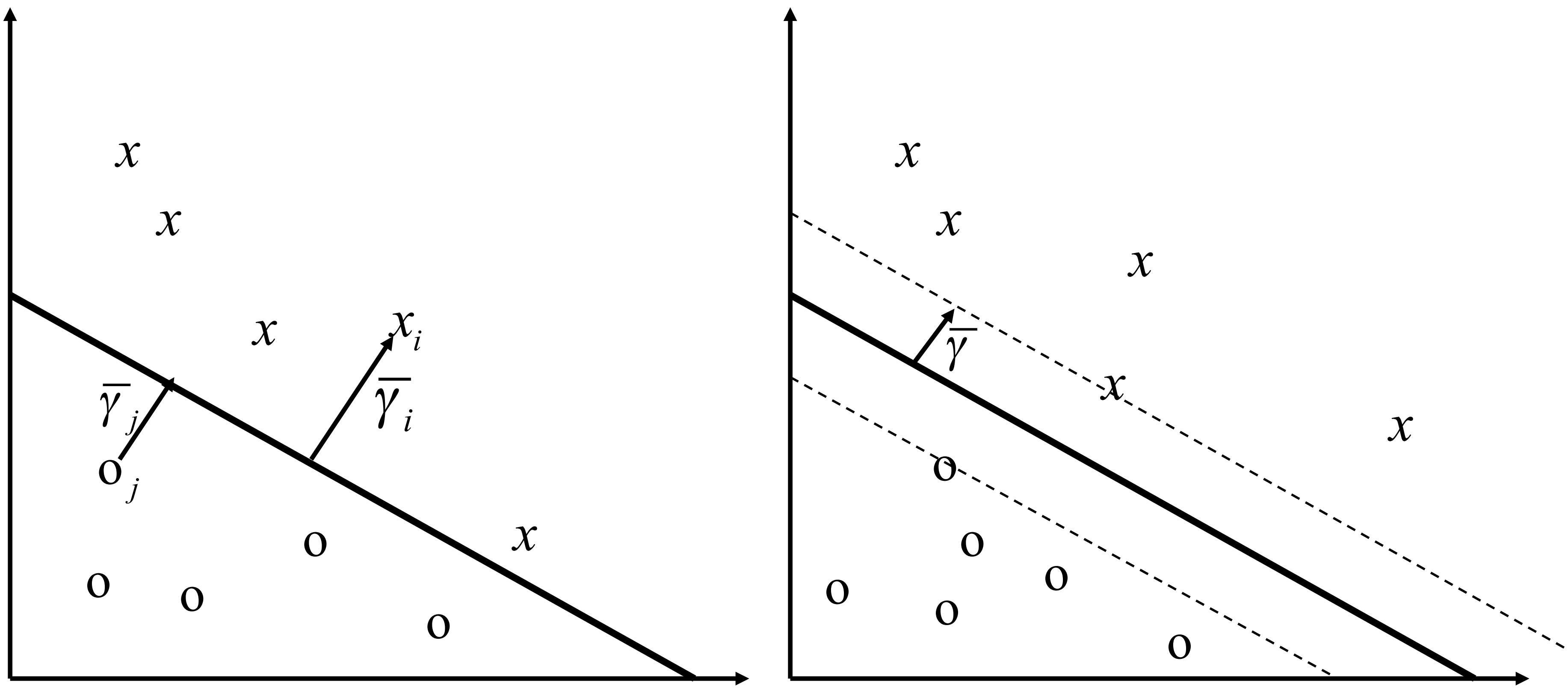
$$\|\vec{x}\| \cos(\vec{x}, \vec{w}) = \frac{\vec{x} \cdot \vec{w}}{\|\vec{w}\|} = \vec{x} \cdot \frac{\vec{w}}{\|\vec{w}\|}$$

- Norm of $\vec{x}$ times the cosine between $\vec{x}$ and $\vec{w}$, i.e. the projection of $\vec{x}$ on $\vec{w}$

Geometric Margin



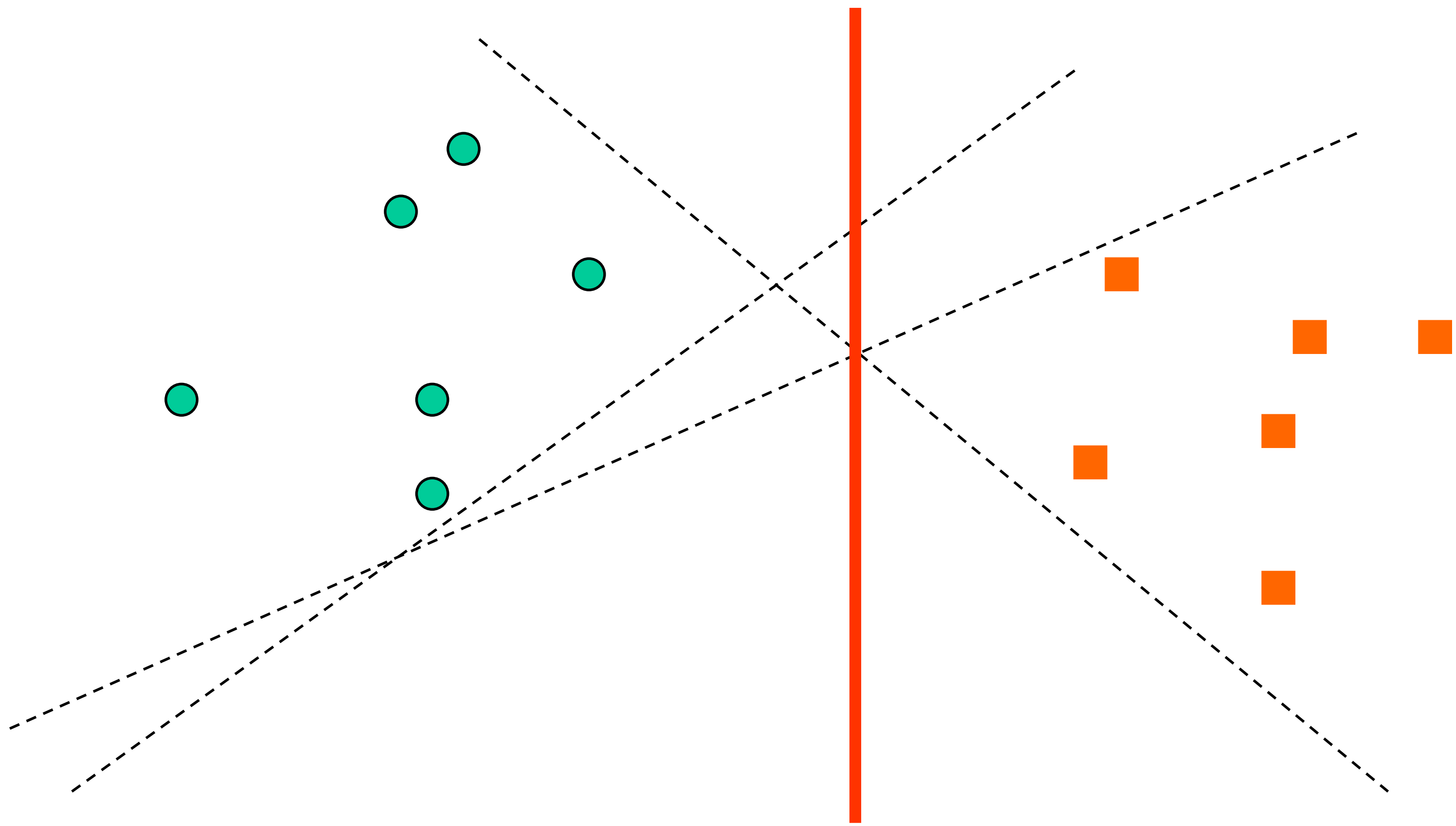
Geometric margins of 2 points and hyperplane margin



Geometric Margin

Hyperplane Margin

Maximal margin vs other margins



Perceptron training on a data set (on-line algorthm)

$\vec{w}_0 \leftarrow \vec{0}; b_0 \leftarrow 0; k \leftarrow 0; R \leftarrow \max_{1 \le i \le l} \| \vec{x}_i \|$

Repeat

for i = 1 to m

if $y_i(\vec{w}_k \cdot \vec{x}_i + b_k) \le 0$ then

$\vec{w}_{k+1} = \vec{w}_k + \eta y_i \vec{x}_i$

$b_{k+1} = b_k + \eta y_i R^2$

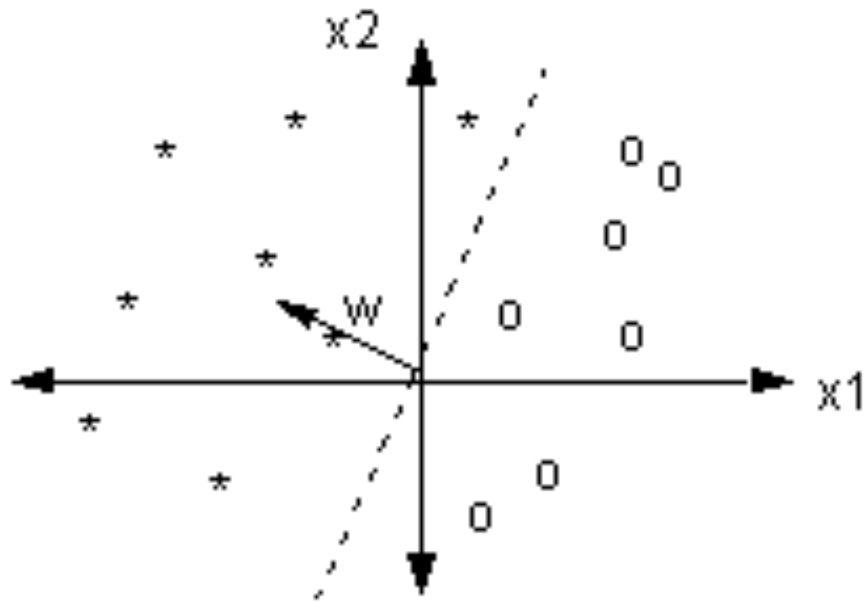
k = k + 1

endif

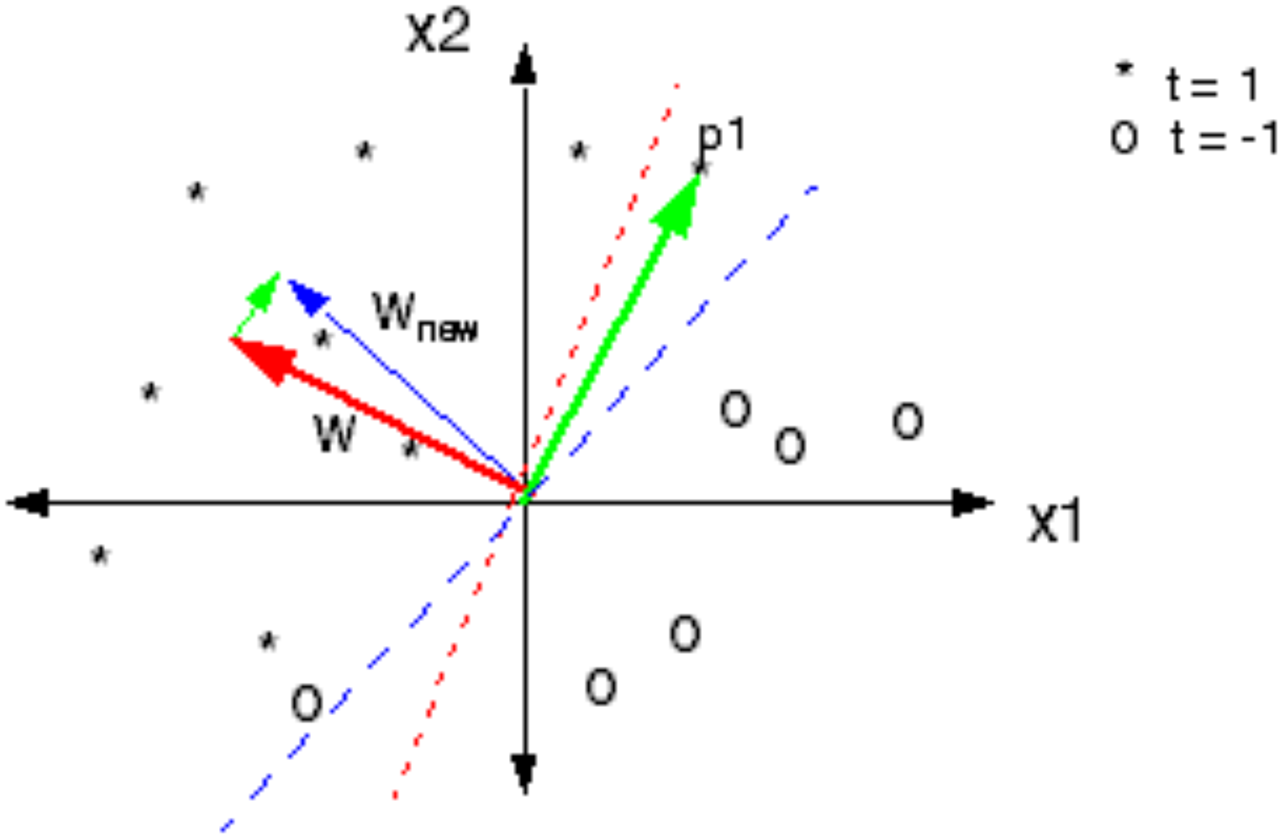
endfor

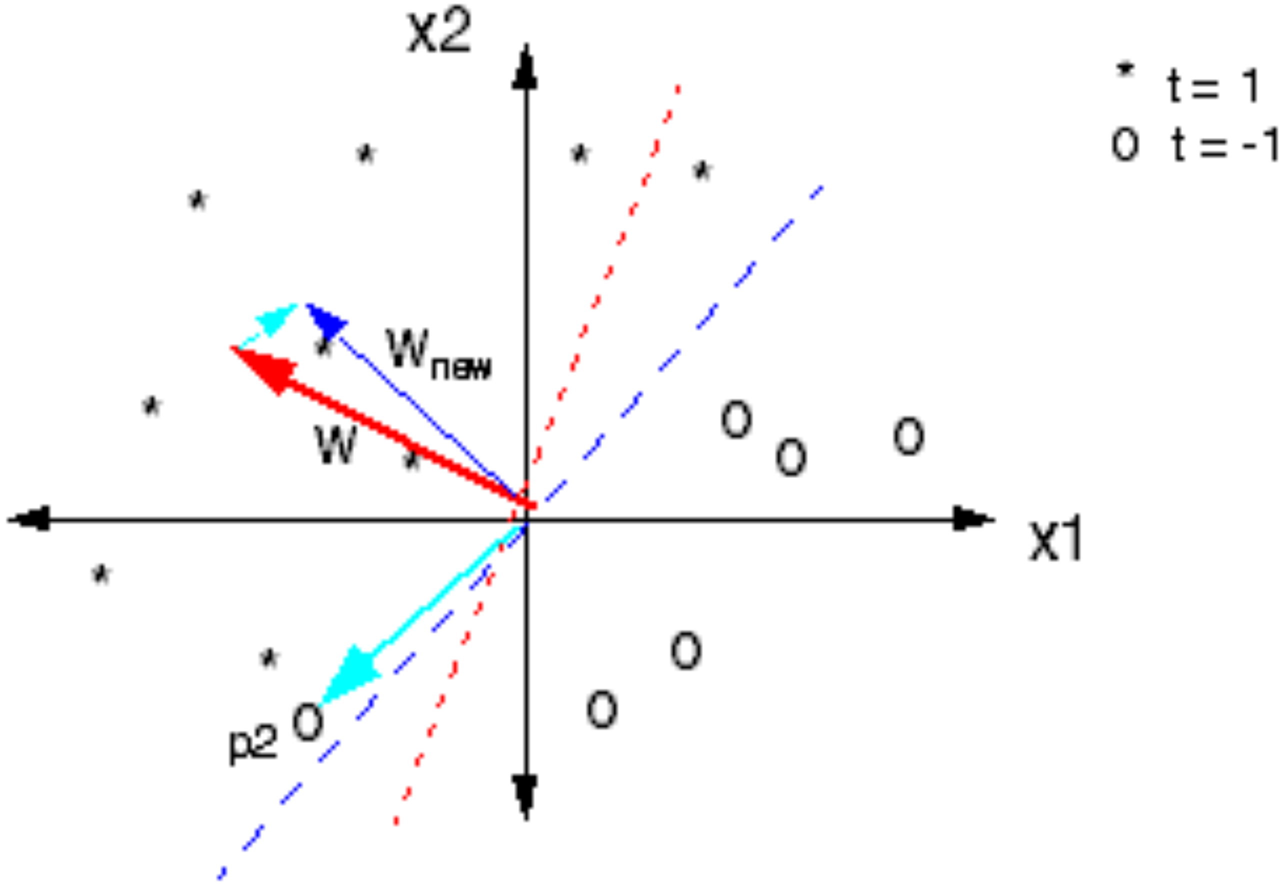
until no error is found

return k,$(\vec{w}_k, b_k)$

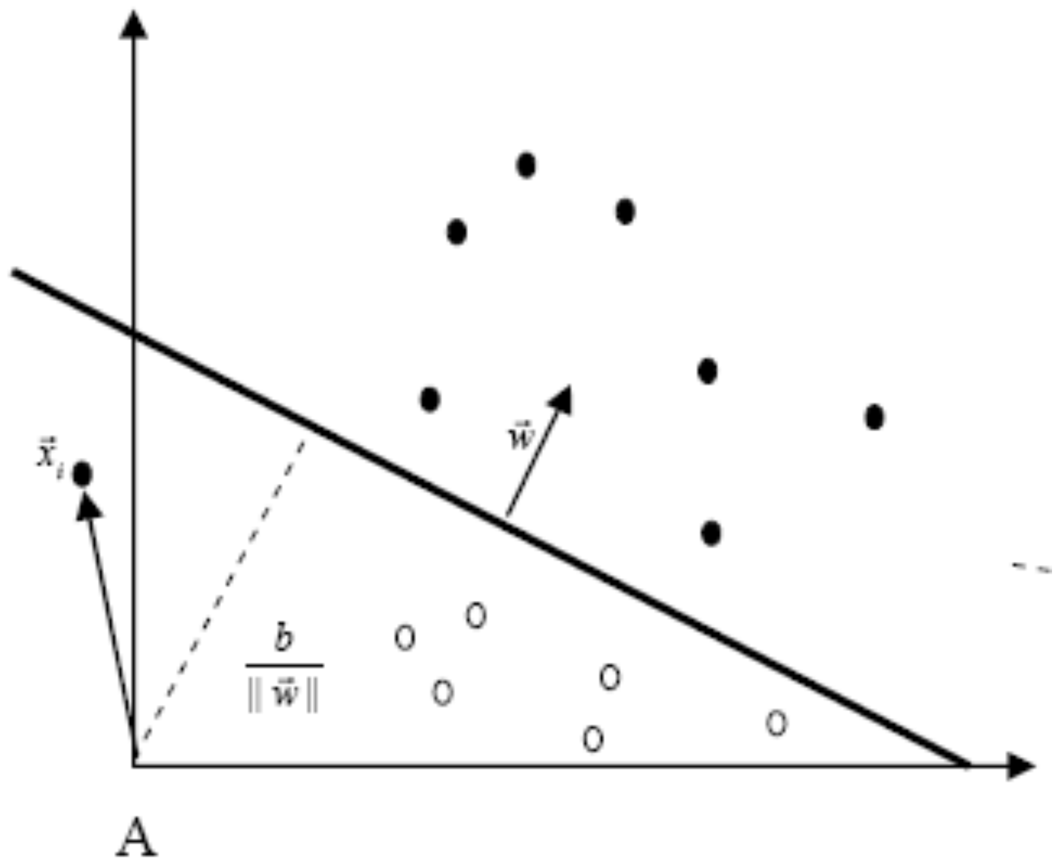


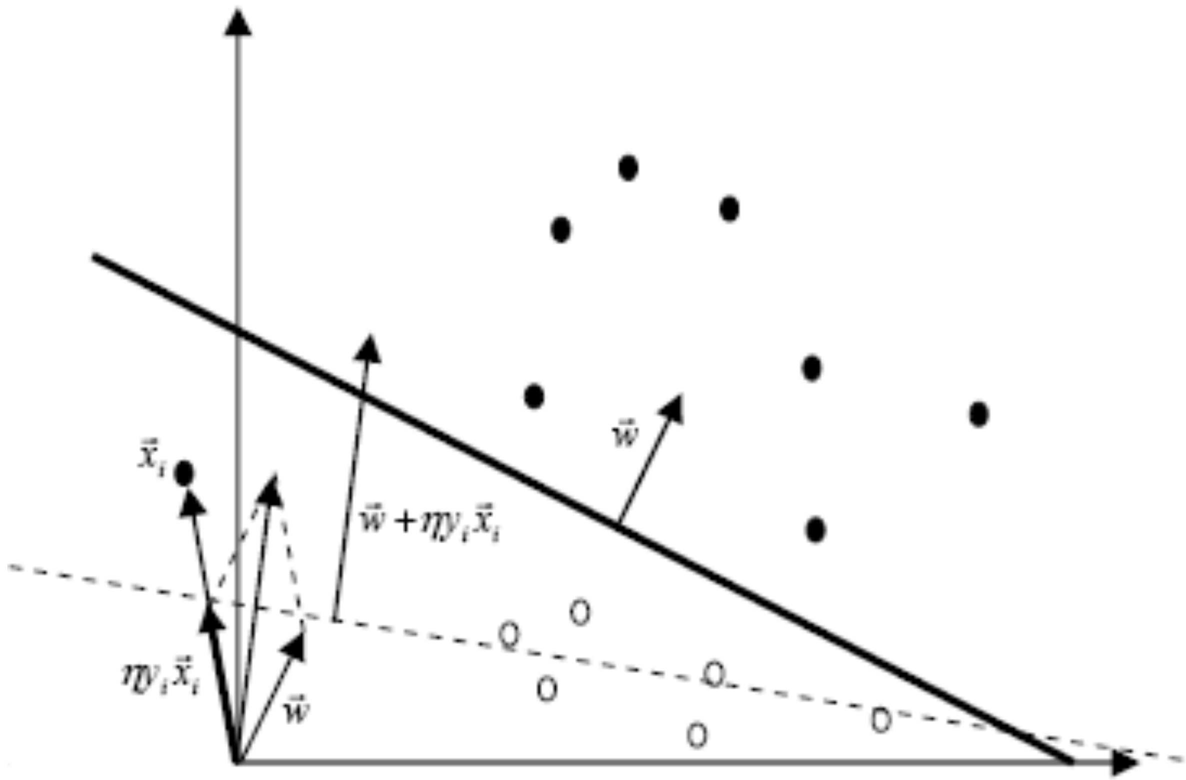
x2
w
x1


x2
x1
p1
W
W_{new}
* t = 1
o t = -1

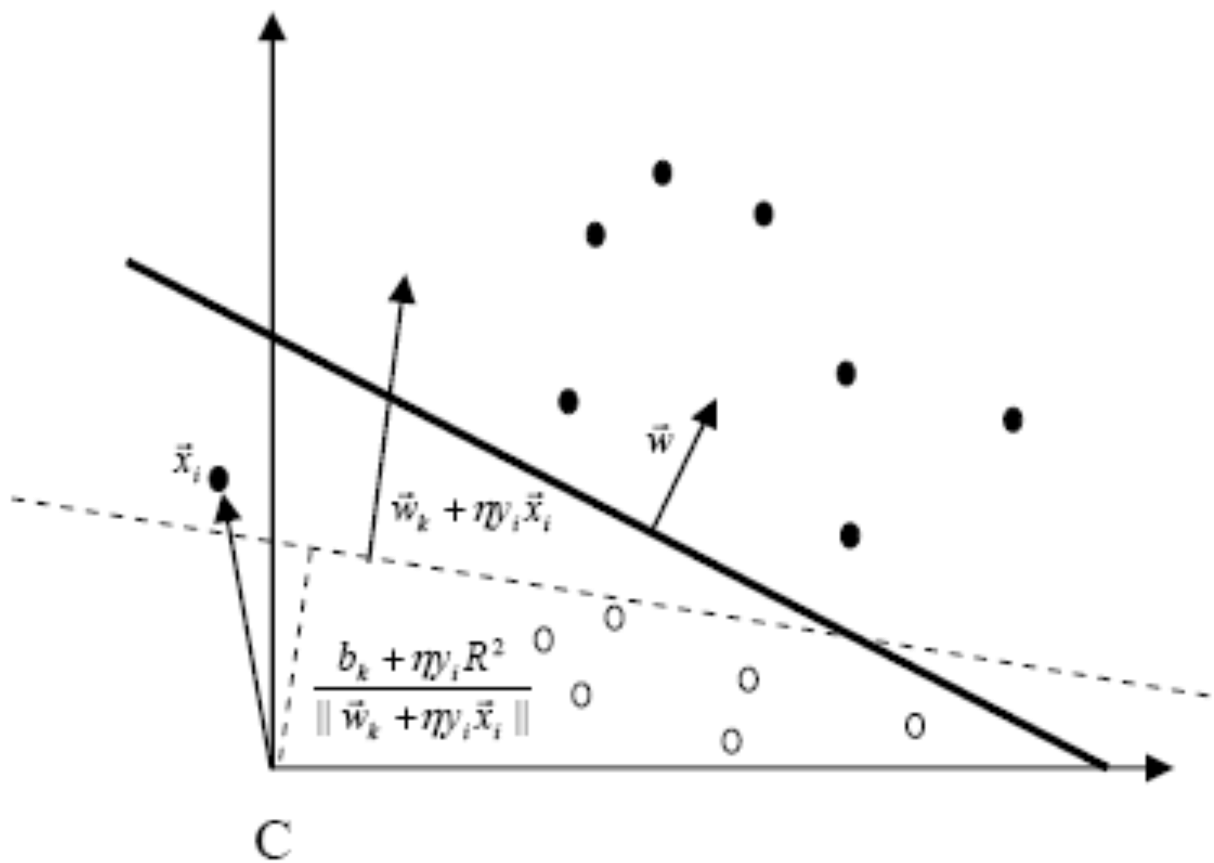


x2
* t = 1
o t = -1
W_new
W
x1
p2


$\vec{x}_i$
$\vec{w}$
$\frac{b}{\|\vec{w}\|}$
A



B



$\vec{x}_i$
$\vec{w}$
$\vec{w}_k + \eta y_i \vec{x}_i$
$\frac{b_k + \eta y_i R^2}{\| \vec{w}_k + \eta y_i \vec{x}_i \|}$
C

Novikoff's Theorem

Let S be a non-trivial training-set and let

$$R = \max_{i=1,..,m} \| x_i \| .$$

Let us suppose there is a vector $\mathbf{w}^*, \| \mathbf{w}^* \| = 1$ and

$$y_i(\langle \mathbf{w}^*, \mathbf{x}_i \rangle + b^*) \geq \gamma, \quad i = 1,..,m,$$

with $\gamma > 0$. Then the maximum number of errors of the perceptron is:

$$t^* = \left(\frac{2R}{\gamma} \right)^2 ,$$

Observations

- The theorem states that independently of the margin size, if data is linearly separable the perceptron algorithm finds the solution in a finite amount of steps.
- This number is inversely proportional to the square of the margin.
- The bound is invariant with respect to the scale of the *patterns* (i.e. only the relative distances count).
- The learning rate is not essential for the convergence.

Dual Representation

- The decision function can be rewritten as:

$$h(x) = \mathrm{sgn}(\vec{w} \cdot \vec{x} + b) = \mathrm{sgn}\Big(\sum_{j=1..m} \alpha_j y_j \vec{x}_j \cdot \vec{x} + b \Big) =$$

$$\mathrm{sgn}\Big(\sum_{i=1..m} \alpha_j y_j \vec{x}_j \cdot \vec{x} + b \Big)$$

- as well as the updating function

$$\text{if } y_i \Big(\sum_{j=1..m} \alpha_j y_j \vec{x}_j \cdot \vec{x}_i + b \Big) \leq 0 \text{ then } \alpha_i = \alpha_i + \eta$$

- The learning rate η only affects the re-scaling of the hyperplane, it does not affect the algorithm, so we can fix $\eta = 1$.

First properties of SVMs

- DUALITY is the first feature of Support Vector Machines
- SVMs are learning machines using the following function:

$$f(x) = \mathrm{sgn}(\vec{w} \cdot \vec{x} + b) = \mathrm{sgn}\left(\sum_{j=1..m} \alpha_j y_j \vec{x}_j \cdot \vec{x} + b\right)$$

- Note that data appears only as scalar product (for both testing and learning phases)
- The Matrix $G = \left(\vec{x}_i \cdot \vec{x}_j\right)_{i,j=1}^{m}$ is called Gram matrix

Limits of Linear Classifiers

- Data must be linearly separable
- Noise (almost all classifier types)
- Data must be in vectorial format

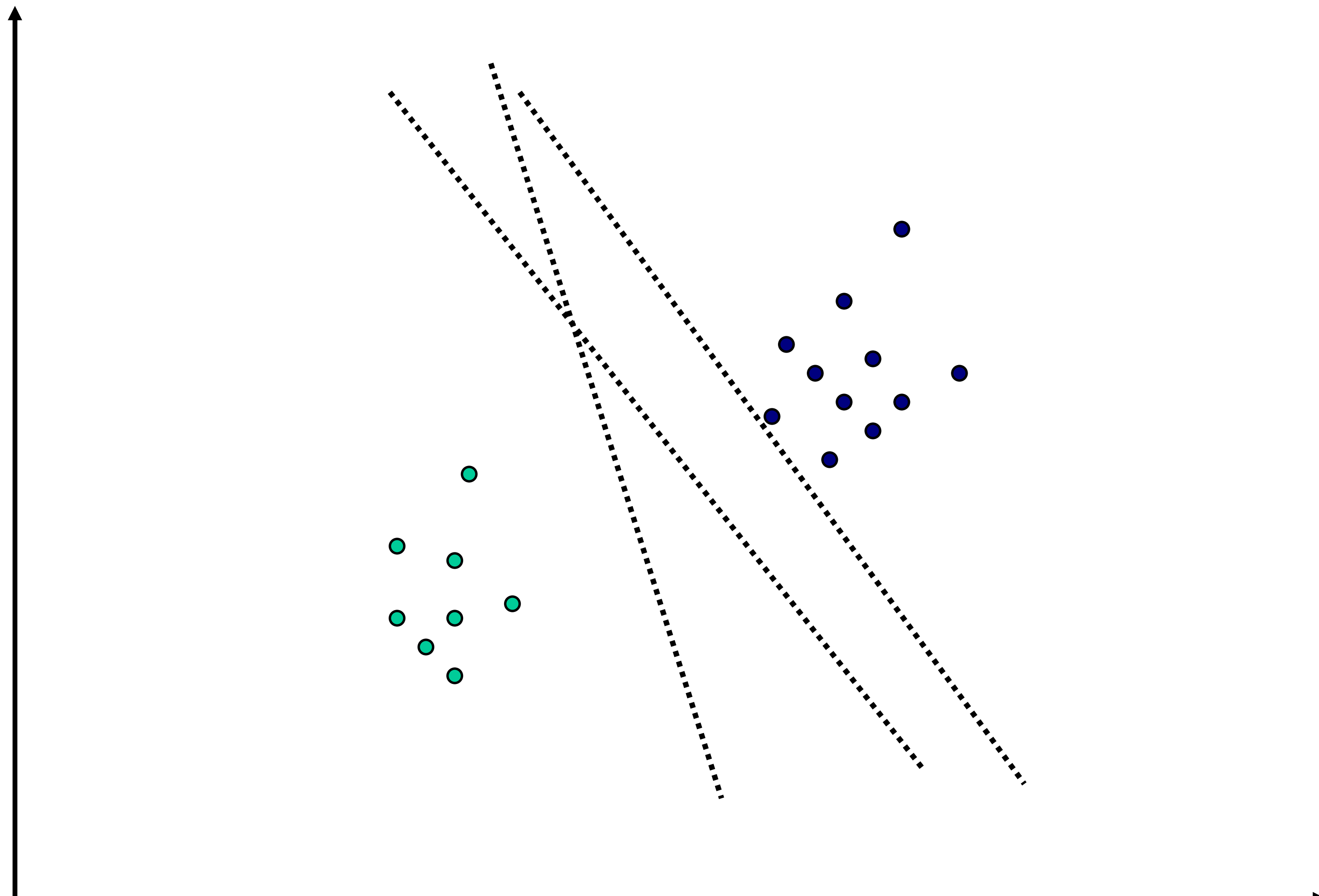
Solutions

- Multi-Layers Neural Network: back-propagation learning algorithm.
- SVMs: kernel methods.

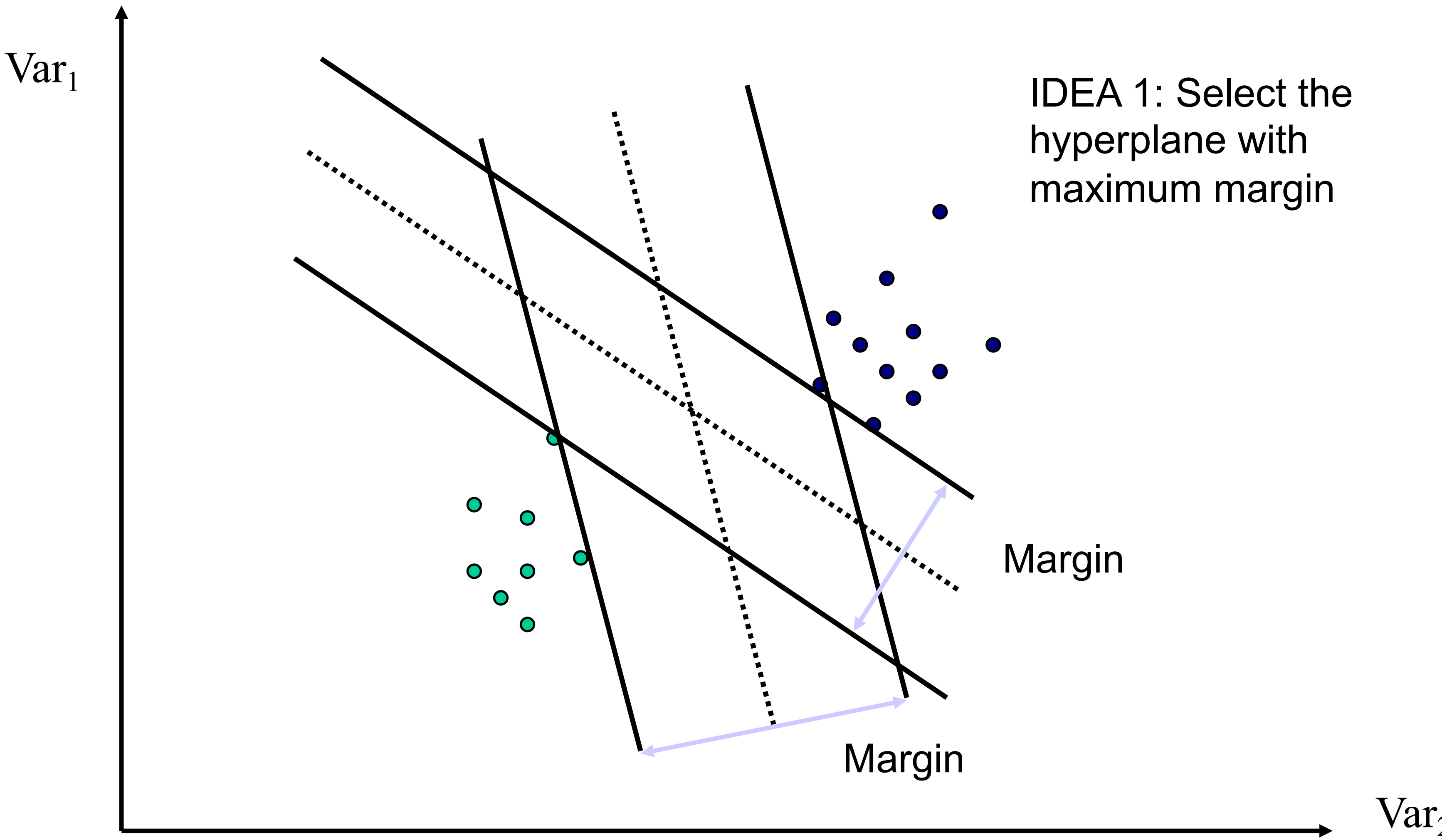
  The learning algorithm is decoupled by the application domain which is encoded by a kernel function

Support Vector Machines

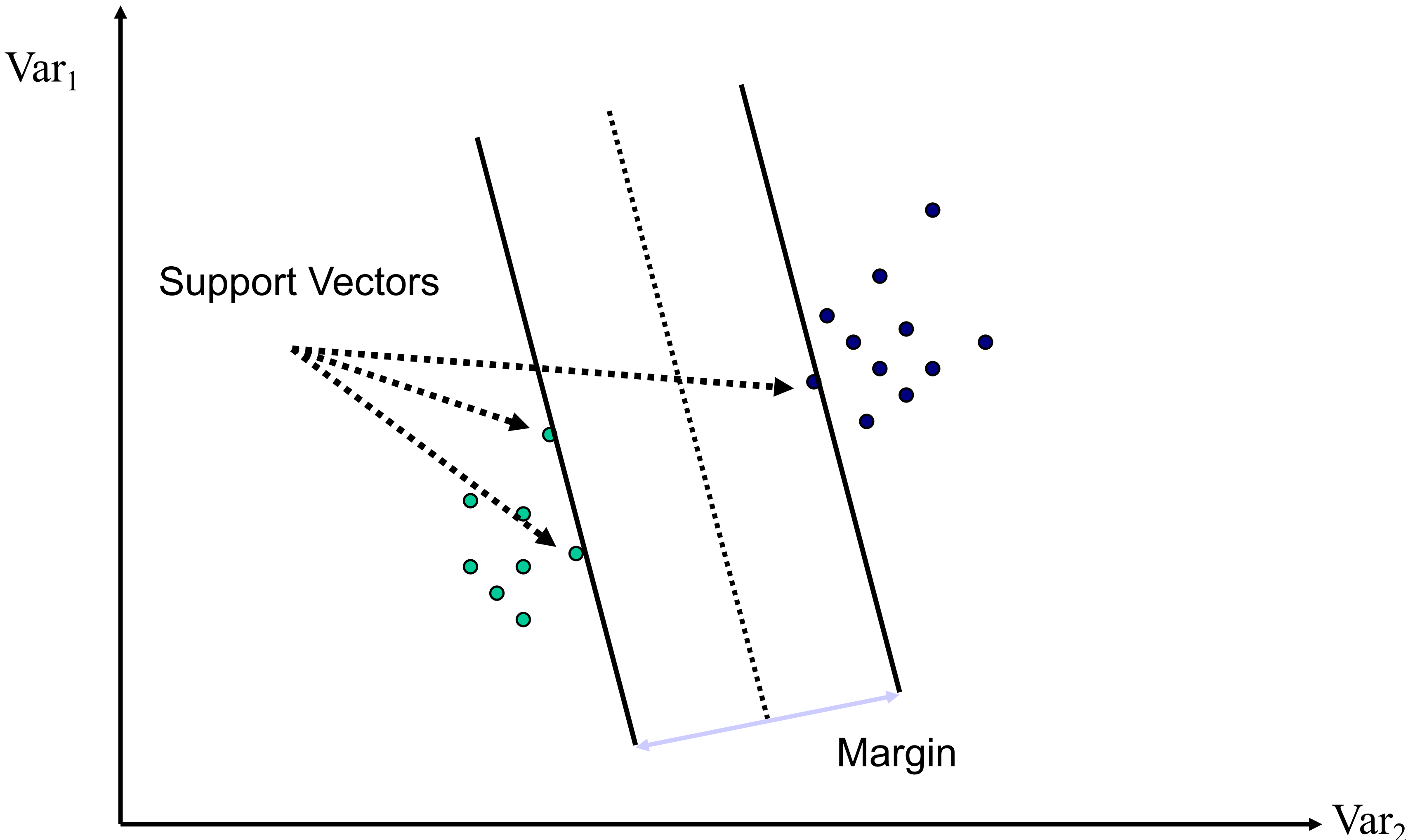
Which hyperplane choose?



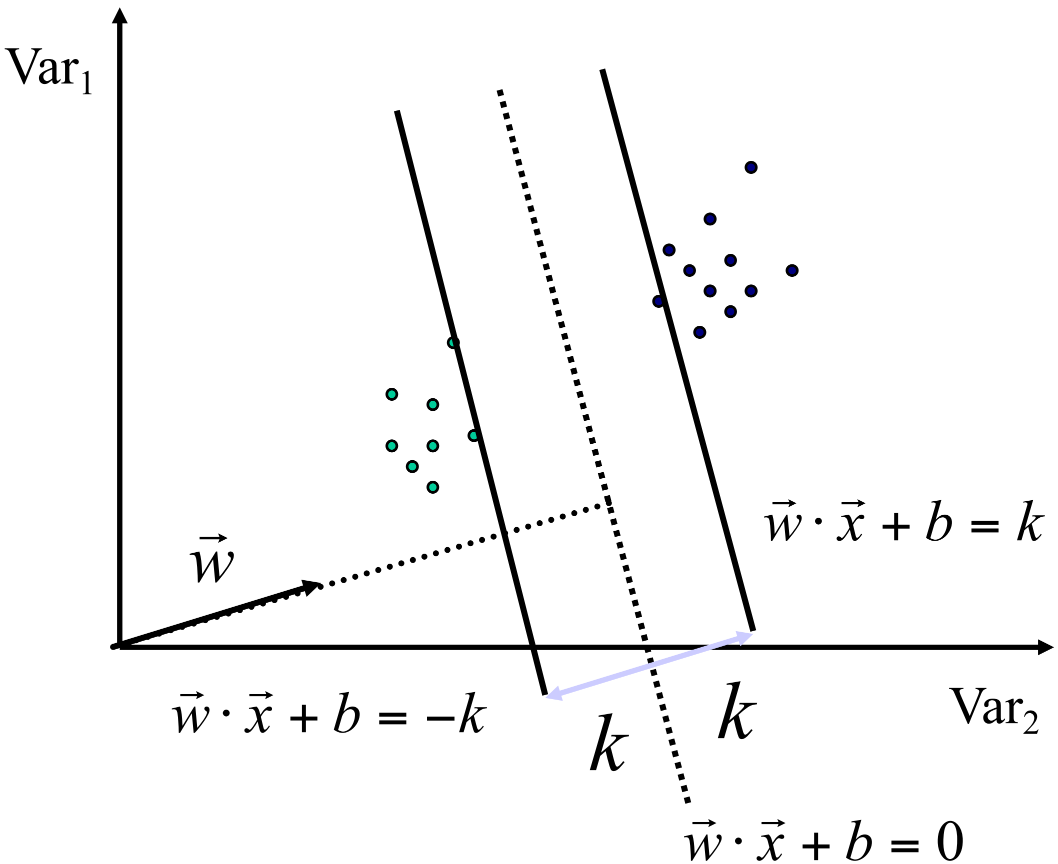
Classifier with a Maximum Margin



Support Vector

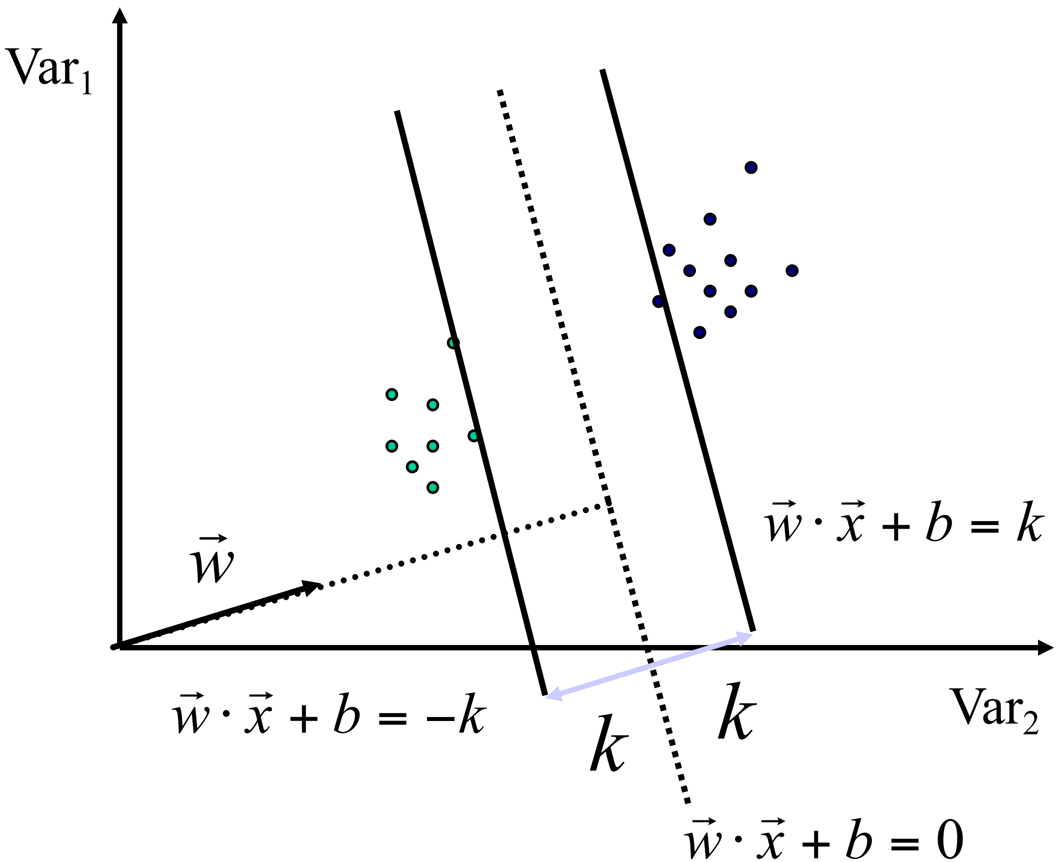


Support Vector Machine Classifiers



The margin is equal to $\frac{2|k|}{\|w\|}$

Support Vector Machines



The margin is equal to $\dfrac{2|k|}{\|w\|}$

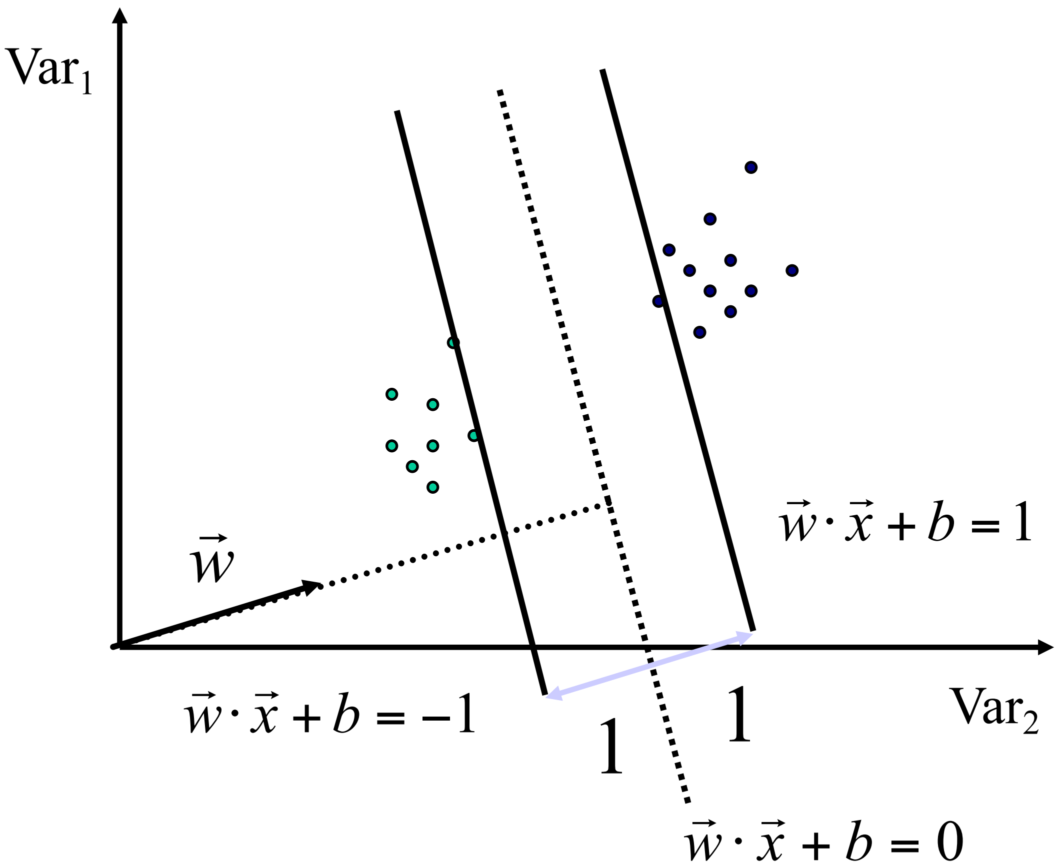
We need to solve

$$\max \frac{2|k|}{\|\vec{w}\|}$$

$$\vec{w} \cdot \vec{x} + b \geq +k, \quad \text{if } \vec{x} \text{ is positive}$$

$$\vec{w} \cdot \vec{x} + b \leq -k, \quad \text{if } \vec{x} \text{ is negative}$$

Support Vector Machines



There is a scale for which *k=1*.

The problem transforms in:

$$\max \frac{2}{\|\vec{w}\|}$$

$$\vec{w}\cdot\vec{x}+b \geq +1, \text{ if } \vec{x} \text{ is positive}$$

$$\vec{w}\cdot\vec{x}+b \leq -1, \text{ if } \vec{x} \text{ is negative}$$

Final Formulation

$$\max \frac{2}{\|\vec{w}\|}$$
$$\vec{w}\cdot\vec{x}_i + b \geq +1, \quad y_i = 1$$
$$\vec{w}\cdot\vec{x}_i + b \leq -1, \quad y_i = -1$$

$$\Rightarrow \quad \max \frac{2}{\|\vec{w}\|} \quad y_i(\vec{w}\cdot\vec{x}_i + b) \geq 1 \quad \Rightarrow$$

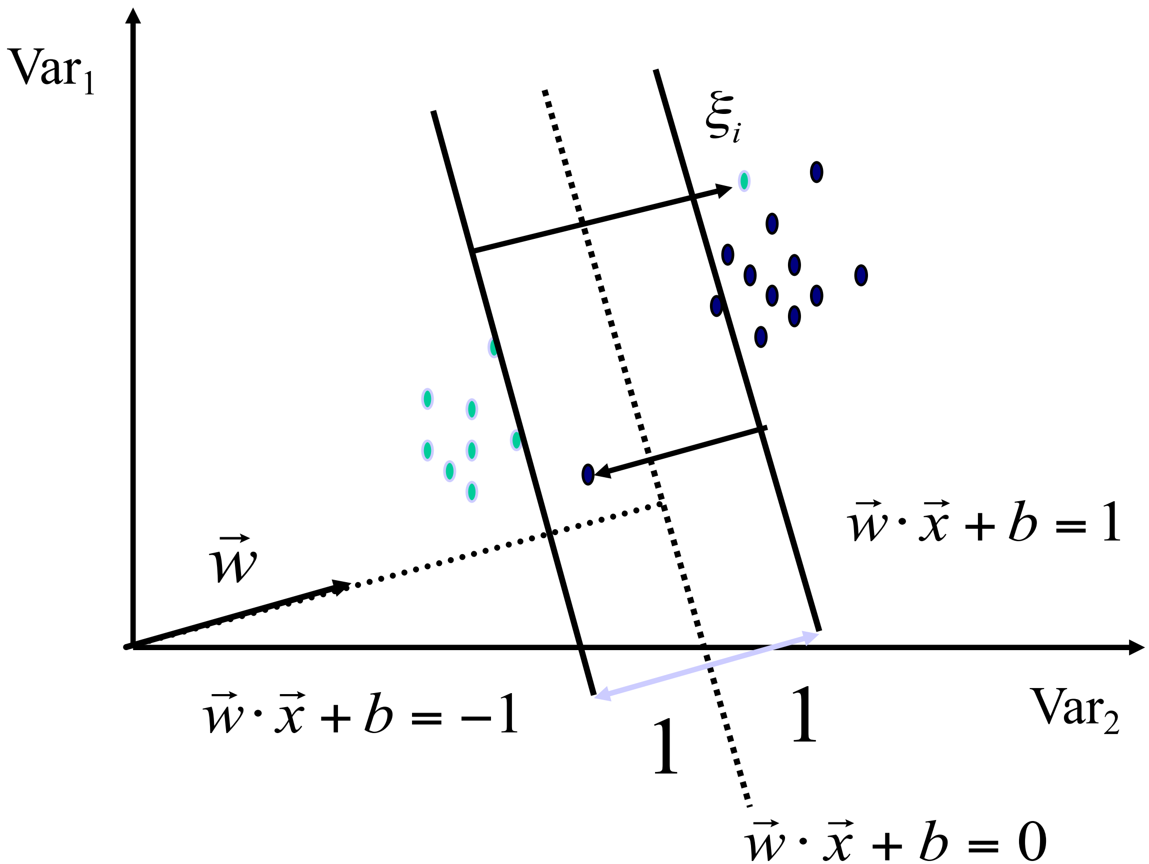
$$\Rightarrow \quad \min \frac{\|\vec{w}\|}{2} \quad y_i(\vec{w}\cdot\vec{x}_i + b) \geq 1$$

$$\Rightarrow \quad \min \frac{\|\vec{w}\|^2}{2} \quad y_i(\vec{w}\cdot\vec{x}_i + b) \geq 1$$

Optimization Problem

- Optimal Hyperplane:
  - Minimize $\tau(\vec{w}) = \frac{1}{2}\|\vec{w}\|^2$
  - Subject to $y_i((\vec{w} \cdot \vec{x}_i) + b) \geq 1, i = 1,\ldots,m$
- The dual problem is simpler

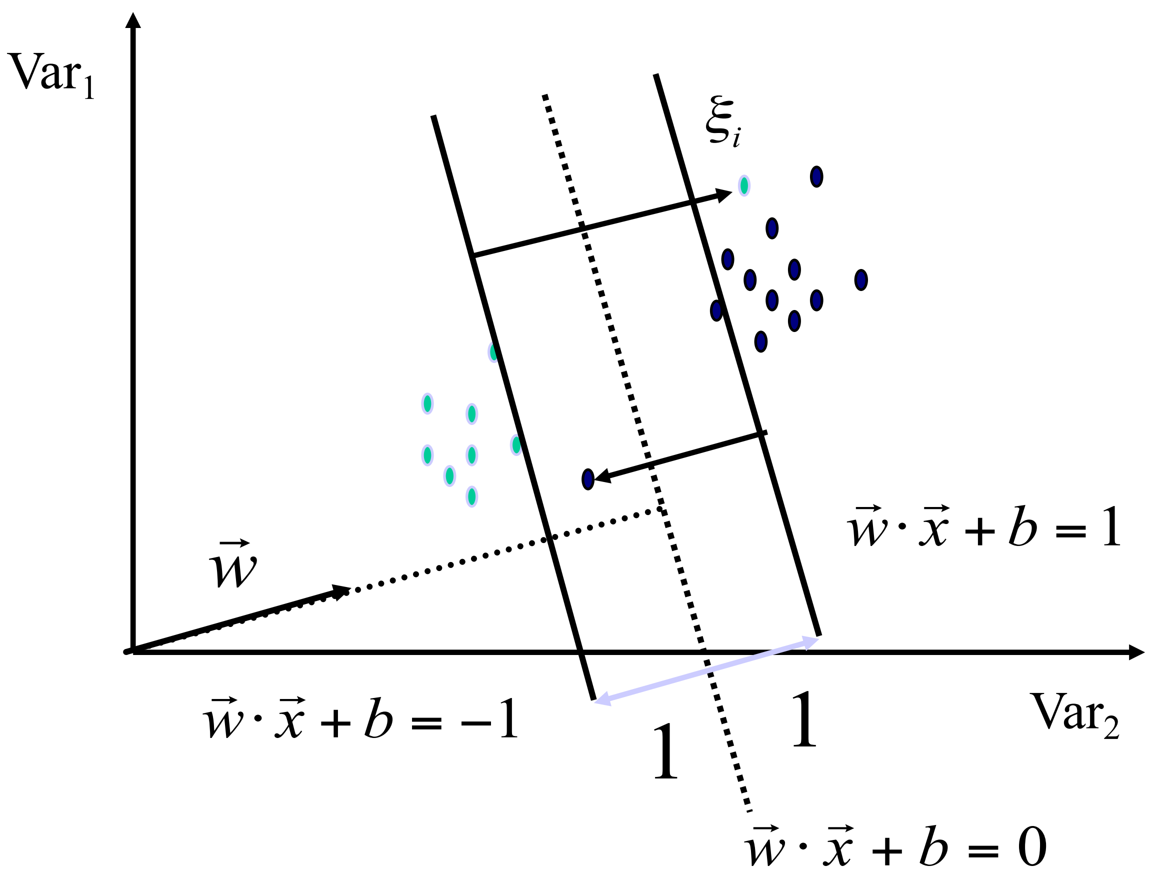
Soft Margin SVMs



ξ_i slack variables are added

Some errors are allowed but they should penalize the objective function

Soft Margin SVMs



The new constraints are

$$y_i(\vec{w} \cdot \vec{x}_i + b) \geq 1 - \xi_i$$
$$\forall \vec{x}_i \quad \text{where} \quad \xi_i \geq 0$$

The objective function penalizes the incorrect classified examples

$$\min \frac{1}{2} \|\vec{w}\|^2 + C \sum_i \xi_i$$

C is the trade-off between margin and the error

Dual formulation

$$\begin{cases} min \quad \frac{1}{2}||\vec{w}|| + C\sum_{i=1}^{m} \xi_i^2 \\ y_i(\vec{w}\cdot\vec{x_i}+b) \geq 1-\xi_i, \quad \forall i=1,..,m \\ \xi_i \geq 0, \quad i=1,..,m \end{cases}$$

$$L(\vec{w},b,\vec{\xi},\vec{\alpha}) = \frac{1}{2}\vec{w}\cdot\vec{w} + \frac{C}{2}\sum_{i=1}^{m}\xi_i^2 - \sum_{i=1}^{m}\alpha_i[y_i(\vec{w}\cdot\vec{x_i}+b)-1+\xi_i],$$

- By deriving wrt $\vec{w}$,$\vec{\xi}$ and b

Final dual optimization problem

$$\sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y_i y_j \alpha_i \alpha_j \left(\vec{x_i} \cdot \vec{x_j} + \frac{1}{C}\delta_{ij}\right)$$

$\alpha_i \geq 0, \quad \forall i = 1, .., m$

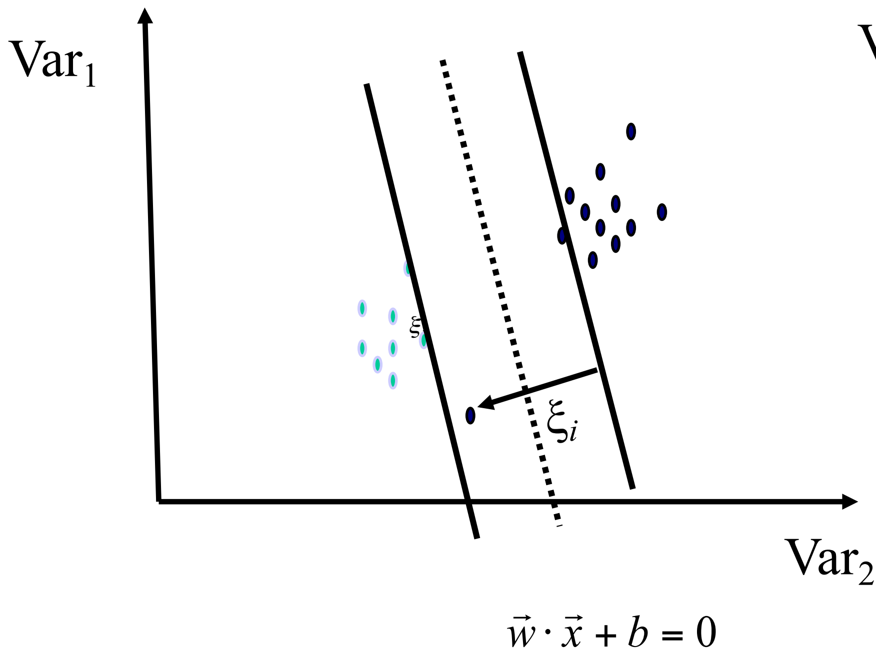
$\sum_{i=1}^{m} y_i \alpha_i = 0$

Soft Margin Support Vector Machines

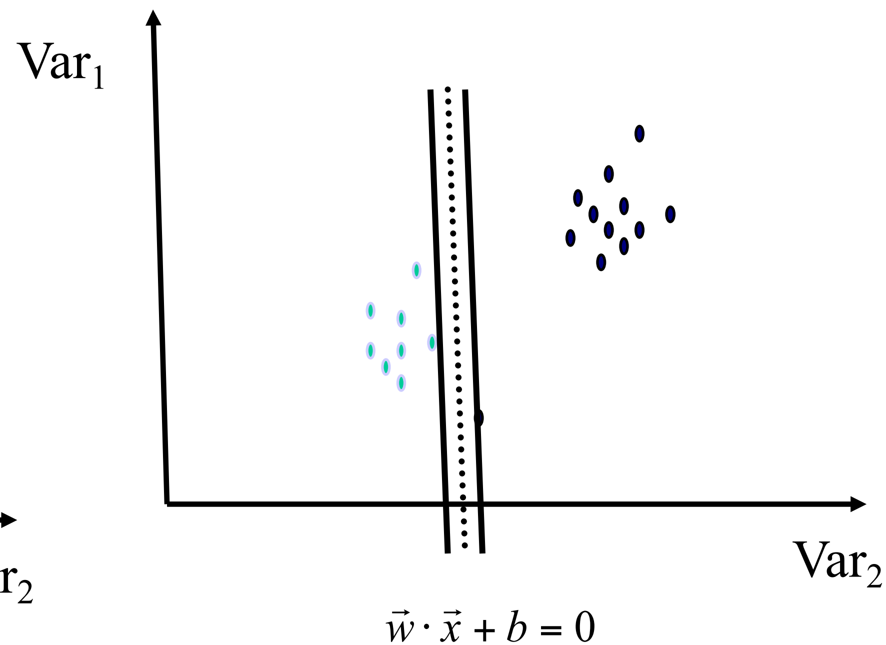
$$\min \frac{1}{2} \|\vec{w}\|^2 + C\sum_i \xi_i \qquad \begin{array}{l} y_i(\vec{w}\cdot\vec{x}_i + b) \ge 1 - \xi_i \quad \forall \vec{x}_i \\ \xi_i \ge 0 \end{array}$$

- The algorithm tries to keep ξ_i low and maximize the margin
- NB: The number of error is not directly minimized (NP-complete problem); the distances from the hyperplane are minimized
- If $C \to \infty$, the solution tends to the one of the *hard-margin* algorithm
- *Attention !!!:* if $C = 0$ we get $\|\vec{w}\| = 0$, since $y_i b \ge 1 - \xi_i \quad \forall \vec{x}_i$
- If C increases the number of error decreases. When C tends to infinite the number of errors must be 0, i.e. the *hard-margin* formulation

Robusteness of *Soft* vs. *Hard Margin* SVMs



Soft Margin SVM



Hard Margin SVM

Soft vs Hard Margin SVMs

- *Soft-Margin* has ever a solution
- Soft-Margin is more robust to odd examples
- *Hard-Margin* does not require parameters

Parameters

$$\min \frac{1}{2} \| \vec{w} \|^2 + C \sum_i \xi_i = \min \frac{1}{2} \| \vec{w} \|^2 + C^+ \sum_i \xi_i^+ + C^- \sum_i \xi_i^-$$

$$= \min \frac{1}{2} \| \vec{w} \|^2 + C \left(J \sum_i \xi_i^+ + \sum_i \xi_i^- \right)$$

- C: trade-off parameter
- J: cost factor

Kernel Methods

An example of kernel-based machine: Perceptron training

$\vec{w}_0 \leftarrow \vec{0}; b_0 \leftarrow 0; k \leftarrow 0; R \leftarrow \max_{1 \le i \le l} \| \vec{x}_i \|$

do

for i = 1 to ℓ

if $y_i(\vec{w}_k \cdot \vec{x}_i + b_k) \le 0$ then

$\vec{w}_{k+1} = \vec{w}_k + \eta y_i \vec{x}_i$

$b_{k+1} = b_k + \eta y_i R^2$

k = k + 1

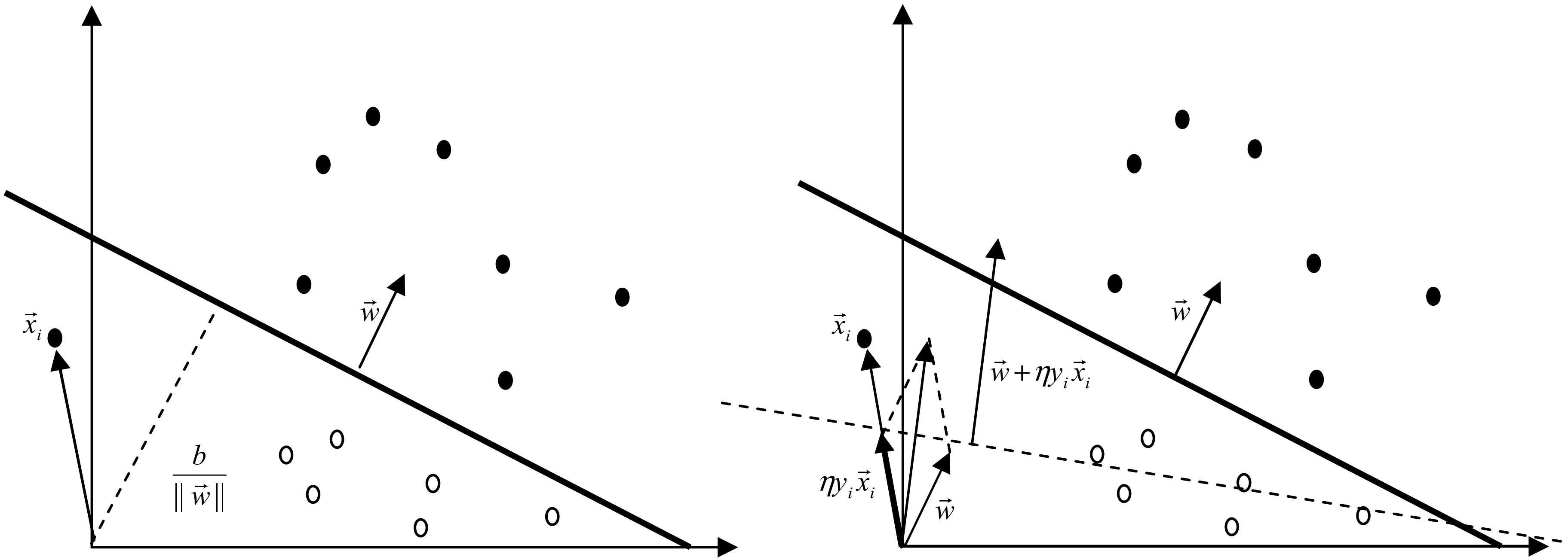
endif

endfor

while an error is found

return k, $(\vec{w}_k, b_k)$

Graphic interpretation of the Perceptron



Dual Representation for Classification

- In each step of perceptron algorithm only training data is added with a certain weight:

$$\vec{w} = \sum_{j=1..\ell} \alpha_j y_j \vec{x}_j$$

- Hence the classification function results:

$$\operatorname{sgn}(\vec{w} \cdot \vec{x} + b) = \operatorname{sgn}\left(\sum_{j=1..\ell} \alpha_j y_j \vec{x}_j \cdot \vec{x} + b \right)$$

- Note that data only appears in the scalar product

Dual Representation for Learning

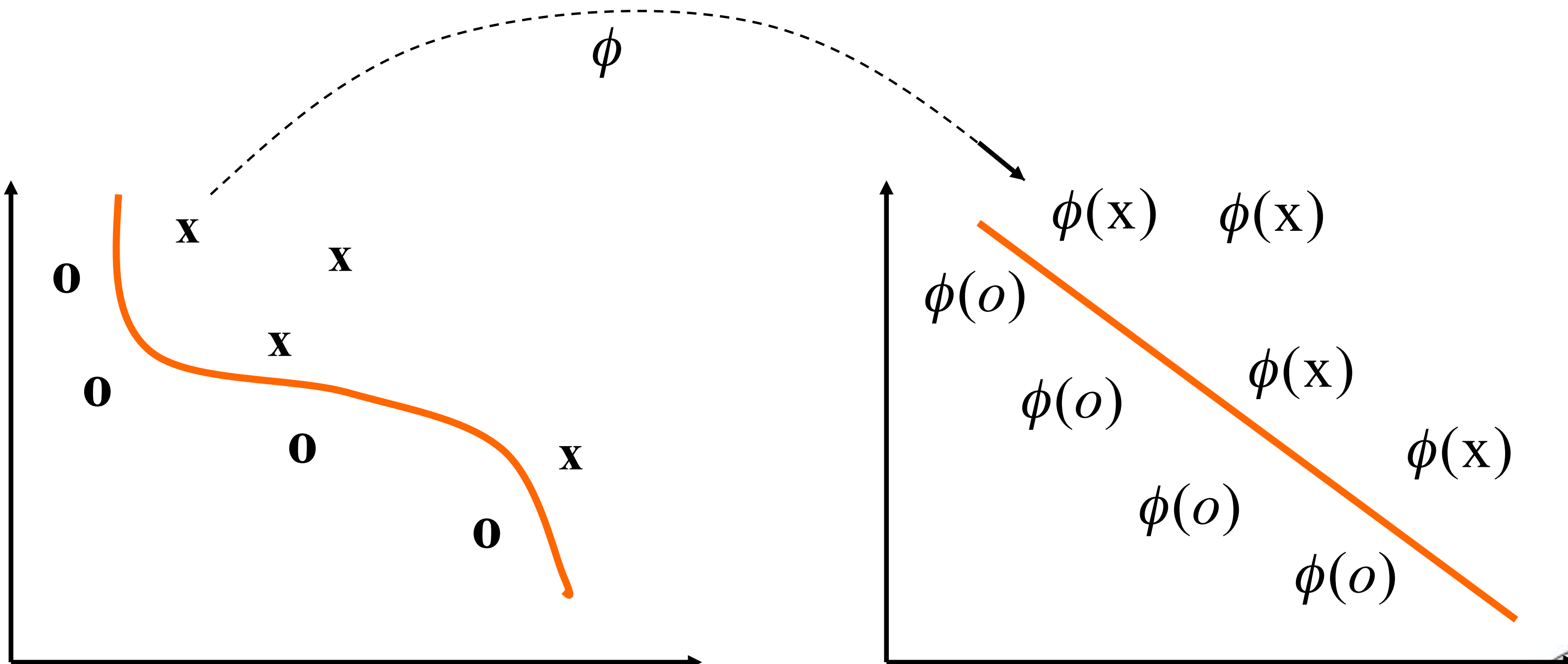
- as well as the updating function

$$\text{if } y_i\left(\sum_{j=1..\ell}\alpha_j y_j \vec{x}_j \cdot \vec{x}_i + b\right) \leq 0 \text{ then } \alpha_i = \alpha_i + \eta$$

- The learning rate η only affects the re-scaling of the hyperplane, it does not affect the algorithm, so we can fix $\eta = 1$

The main idea of Kernel Functions

- Mapping vectors in a space where they are linearly separable, $\vec{x} \rightarrow \phi(\vec{x})$



Soft Margin optimization problem

$$maximize \quad \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{m} y_i y_j \alpha_i \alpha_j \left(\vec{x_i}\cdot\vec{x_j} + \frac{1}{C}\delta_{ij}\right)$$

$$subject \;\; to \quad \alpha_i \geq 0, \quad \forall i = 1,..,m$$

$$\textstyle\sum_{i=1}^{m} y_i \alpha_i = 0$$

Kernels in Support Vector Machines

- In Soft Margin SVMs we maximize:

$$\sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y_i y_j \alpha_i \alpha_j \left(\boldsymbol{x_i} \cdot \boldsymbol{x_j} + \frac{1}{C} \delta_{ij} \right)$$

- By using kernel functions we rewrite the problem as:

$$\begin{cases} maximize \displaystyle\sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y_i y_j \alpha_i \alpha_j \left(k(o_i, o_j) + \frac{1}{C} \delta_{ij} \right) \\ \alpha_i \geq 0, \quad \forall i = 1, .., m \\ \displaystyle\sum_{i=1}^{m} y_i \alpha_i = 0 \end{cases}$$

Kernel Function Definition

Def. 2.26 *A kernel is a function* k, *such that* $\forall\ \vec{x},\vec{z} \in X$

$$k(\vec{x},\vec{z}) = \phi(\vec{x}) \cdot \phi(\vec{z})$$

where ϕ *is a mapping from* X *to an (inner product) feature space.*

- Kernels are the product of mapping functions such as

$$\vec{x} \in \Re^n, \quad \vec{\phi}(\vec{x}) = (\phi_1(\vec{x}), \phi_2(\vec{x}), ..., \phi_m(\vec{x})) \in \Re^m$$

The Kernel Gram Matrix

- The **sole** information used for training is the kernel Gram matrix

$$K_{training} = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & k(\mathbf{x}_1, \mathbf{x}_2) & \dots & k(\mathbf{x}_1, \mathbf{x}_m) \\ k(\mathbf{x}_2, \mathbf{x}_1) & k(\mathbf{x}_2, \mathbf{x}_2) & \dots & k(\mathbf{x}_2, \mathbf{x}_m) \\ \dots & \dots & \dots & \dots \\ k(\mathbf{x}_m, \mathbf{x}_1) & k(\mathbf{x}_m, \mathbf{x}_2) & \dots & k(\mathbf{x}_m, \mathbf{x}_m) \end{bmatrix}$$

- If the kernel is valid, K is symmetric positive-semidefinite

Valid Kernels

Def. B.11 *Eigen Values*
Given a matrix $\boldsymbol{A} \in \mathbb{R}^m \times \mathbb{R}^n$*, an egeinvalue* λ *and an egeinvector* $\vec{x} \in \mathbb{R}^n - \{\vec{0}\}$ *are such that*

$$\boldsymbol{A}\vec{x} = \lambda\vec{x}$$

Def. B.12 *Symmetric Matrix*
A square matrix $\boldsymbol{A} \in \mathbb{R}^n \times \mathbb{R}^n$ *is symmetric iff* $\boldsymbol{A}_{ij} = \boldsymbol{A}_{ji}$ *for* $i \neq j$ $i = 1, .., m$ *and* $j = 1, .., n$*, i.e. iff* $\boldsymbol{A} = \boldsymbol{A}'$.

Def. B.13 *Positive (Semi-) definite Matrix*
A square matrix $\boldsymbol{A} \in \mathbb{R}^n \times \mathbb{R}^n$ *is said to be positive (semi-) definite if its eigenvalues are all positive (non-negative).*

Valid Kernels cont'd

Proposition 1. *(Mercer's conditions)*
Let X be a finite input space and let $K(\boldsymbol{x}, \boldsymbol{z})$ be a symmetric function on X. Then $K(\boldsymbol{x}, \boldsymbol{z})$ is a kernel function if and only if the matrix

$$k(\boldsymbol{x}, \boldsymbol{z}) = \boldsymbol{\phi}(\boldsymbol{x}) \cdot \boldsymbol{\phi}(\boldsymbol{z})$$

is positive semi-definite (has non-negative eigenvalues).

- If the matrix is positive semi-definite then we can find a mapping ϕ implementing the kernel function

Is it a valid kernel?

- It may not be a kernel so we can use ***M´·M***

Proposition B.14 *Let $\boldsymbol{A}$ be a symmetric matrix. Then A is positive (semi-) definite iff for any vector $\vec{x} \neq 0$*

$$\vec{x}'\boldsymbol{A}\vec{x} > \lambda\vec{x} \quad (\geq 0).$$

From the previous proposition it follows that: If we find a decomposition $\boldsymbol{A}$ in $\boldsymbol{M}'\boldsymbol{M}$, then $\boldsymbol{A}$ is semi-definite positive matrix as

$$\vec{x}'\boldsymbol{A}\vec{x} = \vec{x}'\boldsymbol{M}'\boldsymbol{M}\vec{x} = (\boldsymbol{M}\vec{x})'(\boldsymbol{M}\vec{x}) = \boldsymbol{M}\vec{x} \cdot \boldsymbol{M}\vec{x} = ||\boldsymbol{M}\vec{x}||^2 \geq 0.$$

Valid Kernel operations

- $k(x,z) = k_1(x,z)+k_2(x,z)$
- $k(x,z) = k_1(x,z)*k_2(x,z)$
- $k(x,z) = \alpha\, k_1(x,z)$
- $k(x,z) = f(x)f(z)$
- $k(x,z) = x'Bz$
- $k(x,z) = k_1(\phi(x),\phi(z))$

Object Transformation [Moschitti et al, CLJ 2008]

- $$K(O_1,O_2) = \phi(O_1)\cdot\phi(O_2) = \phi_E(\phi_M(O_1))\cdot\phi_E(\phi_M(O_2))$$
  $$= \phi_E(S_1)\cdot\phi_E(S_2) = K_E(S_1,S_2)$$
- ***Canonical Mapping***, $\phi_M()$
  - object transformation,
  - e. g., a syntactic parse tree into a verb subcategorization frame tree.
- ***Feature Extraction***, $\phi_E()$
  - maps the canonical structure in all its fragments
  - different fragment spaces, e.g. String and Tree Kernels

Part I – Basic Kernels (for structured data)

- Basic Kernels and their Feature Spaces (35 min)
  - Linear Kernels
  - Polynomial Kernels
  - Lexical Semantic Kernels
  - String and Word Sequence Kernels
  - Syntactic Tree Kernel, Partial Tree kernel (PTK), Semantic Syntactic Tree Kernel, Smoothed PTK

Linear Kernel

- In Text Categorization documents are word vectors

$$\Phi(d_x) = \vec{x} = (0,..,1,..,0,..,0,..,1,..,0,..,0,..,1,..,0,..,0,..,1,..,0,..,1)$$

buy market sell stocks trade

$$\Phi(d_z) = \vec{z} = (0,..,1,..,0,..,1,..,0,..,0,..,0,..,1,..,0,..,0,..,1,..,0,..,0)$$

buy company sell stock

- The dot product $\vec{x} \cdot \vec{z}$ counts the number of features in common
- This provides a sort of *similarity*

Feature Conjunction (polynomial kernel)

- The initial vectors are mapped in a higher space

$$\Phi(< x_1, x_2 >) \rightarrow (x_1^2, x_2^2, \sqrt{2}x_1x_2, \sqrt{2}x_1, \sqrt{2}x_2, 1)$$

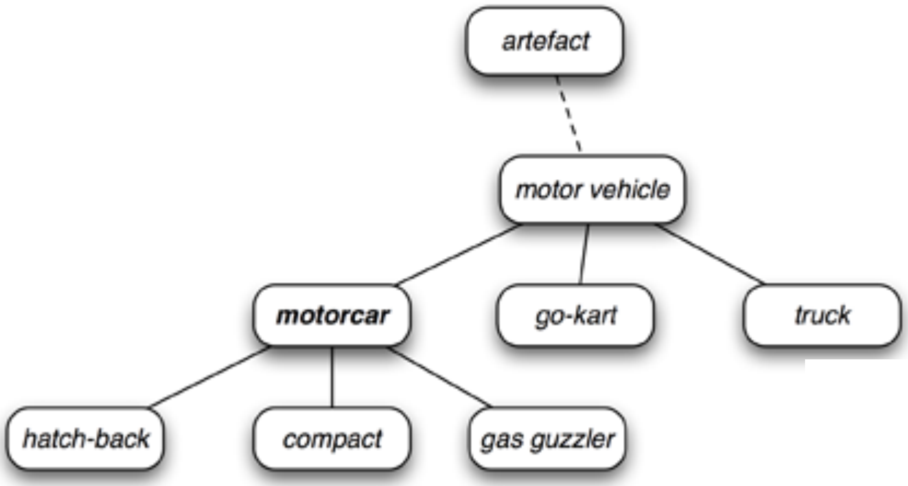
- More expressive, as (x_1x_2) encodes

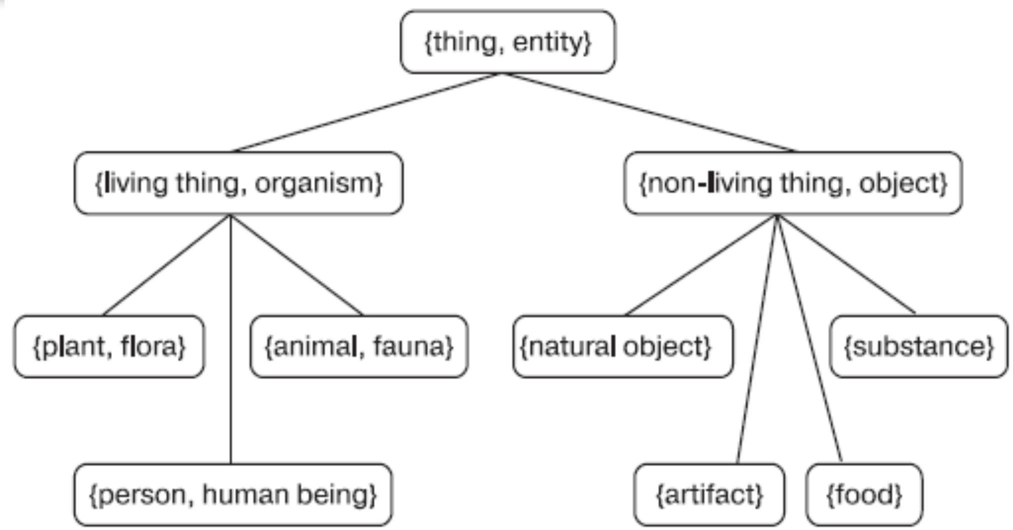
  `Stock+Market` vs. **`Downtown+Market`** features

- We can smartly compute the scalar product as

$$\Phi(\vec{x}) \cdot \Phi(\vec{z}) =$$
$$= (x_1^2, x_2^2, \sqrt{2}x_1x_2, \sqrt{2}x_1, \sqrt{2}x_2, 1) \cdot (z_1^2, z_2^2, \sqrt{2}z_1z_2, \sqrt{2}z_1, \sqrt{2}z_2, 1) =$$
$$= x_1^2z_1^2 + x_2^2z_2^2 + 2x_1x_2z_1z_2 + 2x_1z_1 + 2x_2z_2 + 1 =$$
$$= (x_1z_1 + x_2z_2 + 1)^2 = (\vec{x} \cdot \vec{z} + 1)^2 = K_{Poly}(\vec{x}, \vec{z})$$

Sub-hierarchies in WordNet





Similarity based on WordNet

Inverted Path Length:

$$sim_{IPL}(c_1, c_2) = \frac{1}{(1 + d(c_1, c_2))^{\alpha}}$$

Wu & Palmer:

$$sim_{WUP}(c_1, c_2) = \frac{2\,dep(lso(c_1, c_2))}{d(c_1, lso(c_1, c_2)) + d(c_2, lso(c_1, c_2)) + 2\,dep(lso(c_1, c_2))}$$

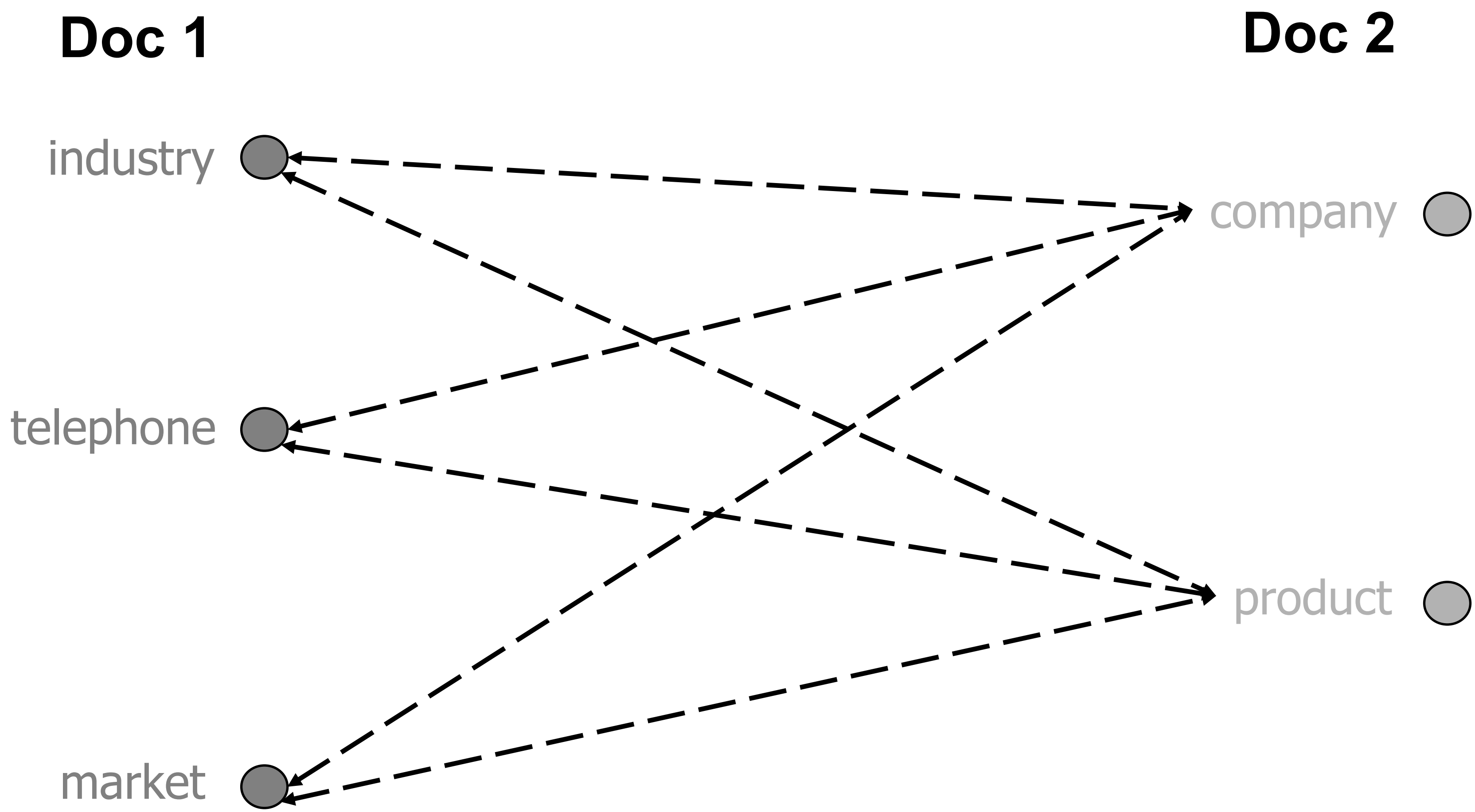
Resnik:

$$sim_{RES}(c_1, c_2) = -\log\ P(lso(c_1, c_2))$$

Lin:

$$sim_{LIN}(c_1, c_2) = \frac{2\ \log\ P(lso(c_1, c_2))}{\log\ P(c_1) + \log P\,(c_2)}$$

Document Similarity



Lexical Semantic Kernels

- The document similarity is the following SK function:

$$SK(d_1, d_2) = \sum_{w_1 \in d_1, w_2 \in d_2} s(w_1, w_2)$$

- where *s* is any similarity function between words, e.g. WordNet [Basili et al.,2005] similarity or LSA [Cristianini et al., 2002]
- Good results when training data is small

String Kernel

- Given two strings, the number of matches between their substrings is evaluated
- E.g. Bank and Rank
  - B, a, n, k, Ba, Ban, Bank, Bk, an, ank, nk,..
  - R, a , n , k, Ra, Ran, Rank, Rk, an, ank, nk,..
- String kernel over sentences and texts
- Huge space but there are efficient algorithms

Using character sequences

$$\phi("bank") = \vec{x} = (0,..,1,..,0,..,1,..,0,......1,..,0,..,1,..,0,..,1,..,0)$$

bank ank bnk bk b

$$\phi("rank") = \vec{z} = (1,..,0,..,0,..,1,..,0,......0,..,1,..,0,..,1,..,0,..,1)$$

rank ank rnk rk r

- $\vec{x} \cdot \vec{z}$ counts the number of common substrings

$$\vec{x} \cdot \vec{z} = \phi("bank") \cdot \phi("rank") = k("bank","rank")$$

Formal Definition

$$s = s_1, .., s_{|s|}, \quad \vec{I} = (i_1, ..., i_{|u|})$$

$$u = s[\vec{I}]$$

$$\phi_u(s) = \sum_{\vec{I}:u=s[\vec{I}]} \lambda^{l(\vec{I})}, \text{ where } \; l(\vec{I}) = i_{|u|} - i_1 + 1$$

$$K(s,t) = \sum_{u \in \Sigma^*} \phi_u(s) \cdot \phi_u(t) = \sum_{u \in \Sigma^*} \sum_{\vec{I}:u=s[\vec{I}]} \lambda^{l(\vec{I})} \sum_{\vec{J}:u=t[\vec{J}]} \lambda^{l(\vec{J})} =$$

$$= \sum_{u \in \Sigma^*} \sum_{\vec{I}:u=s[\vec{I}]} \sum_{\vec{J}:u=t[\vec{J}]} \lambda^{l(\vec{I})+l(\vec{J})}, \text{ where } \; \Sigma^* = \bigcup_{n=0}^{\infty} \Sigma^n$$

Kernel between Bank and Rank

`B`, `a`, `n`, `k`, `Ba`, `Ban`, `Bank`, `an`, `ank`, `nk`, `Bn`, `Bnk`, `Bk` and `ak` are the substrings of $Bank$.

`R`, `a`, `n`, `k`, `Ra`, `Ran`, `Rank`, `an`, `ank`, `nk`, `Rn`, `Rnk`, `Rk` and `ak` are the substrings of $Rank$.

An example of string kernel computation

- $\phi_{\mathrm{a}}(\mathrm{Bank}) = \phi_{\mathrm{a}}(\mathrm{Rank}) = \lambda^{(i_1 - i_1 + 1)} = \lambda^{(2-2+1)} = \lambda,$
- $\phi_{\mathrm{n}}(\mathrm{Bank}) = \phi_{\mathrm{n}}(\mathrm{Rank}) = \lambda^{(i_1 - i_1 + 1)} = \lambda^{(3-3+1)} = \lambda,$
- $\phi_{\mathrm{k}}(\mathrm{Bank}) = \phi_{\mathrm{k}}(\mathrm{Rank}) = \lambda^{(i_1 - i_1 + 1)} = \lambda^{(4-4+1)} = \lambda,$
- $\phi_{\mathrm{an}}(\mathrm{Bank}) = \phi_{\mathrm{an}}(\mathrm{Rank}) = \lambda^{(i_2 - i_1 + 1)} = \lambda^{(3-2+1)} = \lambda^2,$
- $\phi_{\mathrm{ank}}(\mathrm{Bank}) = \phi_{\mathrm{ank}}(\mathrm{Rank}) = \lambda^{(i_3 - i_1 + 1)} = \lambda^{(4-2+1)} = \lambda^3,$
- $\phi_{\mathrm{nk}}(\mathrm{Bank}) = \phi_{\mathrm{nk}}(\mathrm{Rank}) = \lambda^{(i_2 - i_1 + 1)} = \lambda^{(4-3+1)} = \lambda^2$
- $\phi_{\mathrm{ak}}(\mathrm{Bank}) = \phi_{\mathrm{ak}}(\mathrm{Rank}) = \lambda^{(i_2 - i_1 + 1)} = \lambda^{(4-2+1)} = \lambda^3$

$$K(\mathrm{Bank}, \mathrm{Rank}) = (\lambda, \lambda, \lambda, \lambda^2, \lambda^3, \lambda^2, \lambda^3) \cdot (\lambda, \lambda, \lambda, \lambda^2, \lambda^3, \lambda^2, \lambda^3)$$

$$= 3\lambda^2 + 2\lambda^4 + 2\lambda^6$$

Efficient Evaluation: Intuition

- Dynamic Programming technique over:
  - The size of the two input strings, *m*, *n* and
  - The size of their common substrings, *p*
- Evaluate the spectrum string kernels
  - Substrings of size *p*
- Sum the contribution of the different *p* spectra

Efficient Evaluation

Given two sequences s_1a and s_2b, we define:

$$D_p(|s_1|, |s_2|) = \sum_{i=1}^{|s_1|} \sum_{r=1}^{|s_2|} \lambda^{|s_1|-i+|s_2|-r} \times SK_{p-1}(s_1[1:i], s_2[1:r]),$$

$s_1[1:i]$ and $s_2[1:r]$ are their subsequences from 1 to i and 1 to r.

$$SK_p(s_1a, s_2b) = \begin{cases} \lambda^2 \times D_p(|s_1|, |s_2|) \text{ if } a = b; \\ 0 \qquad\qquad\qquad\qquad \textit{otherwise}. \end{cases}$$

D_p satisfies the recursive relation:

$$D_p(k, l) = SK_{p-1}(s_1[1:k], s_2[1:l]) + \lambda D_p(k, l-1) +$$
$$+\lambda D_p(k-1, l) - \lambda^2 D_p(k-1, l-1)$$

Evaluating DP2

- Evaluate the weight of the string of size p in case a character will be matched
- This is done by multiplying the double summation by the number of substrings of size p-1

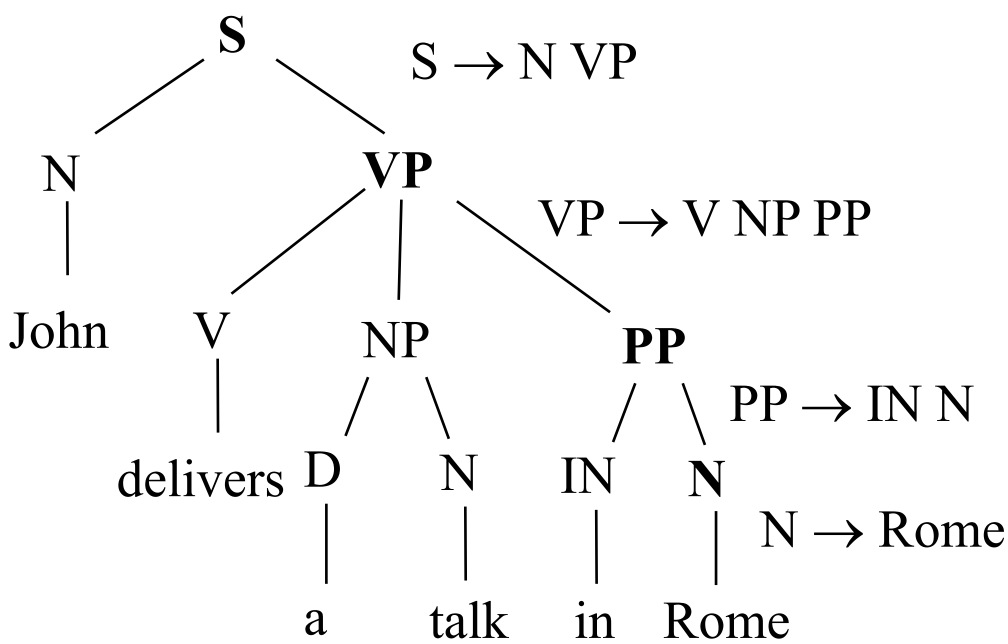
$$D_p(|s_1|, |s_2|) = \sum_{i=1}^{|s_1|} \sum_{r=1}^{|s_2|} \lambda^{|s_1|-i+|s_2|-r} \times SK_{p-1}(s_1[1:i], s_2[1:r])$$

Tree kernels

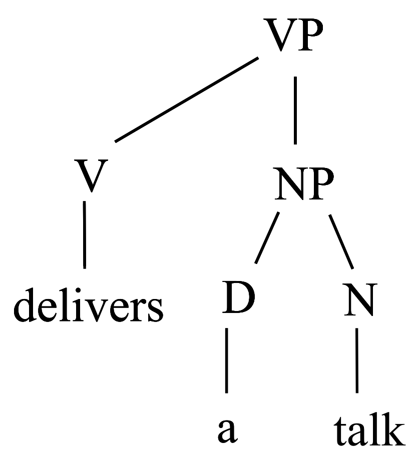
- Syntactic Tree Kernel, Partial Tree kernel (PTK), Semantic Syntactic Tree Kernel, Smoothed PTK
- Efficient computation

Example of a parse tree

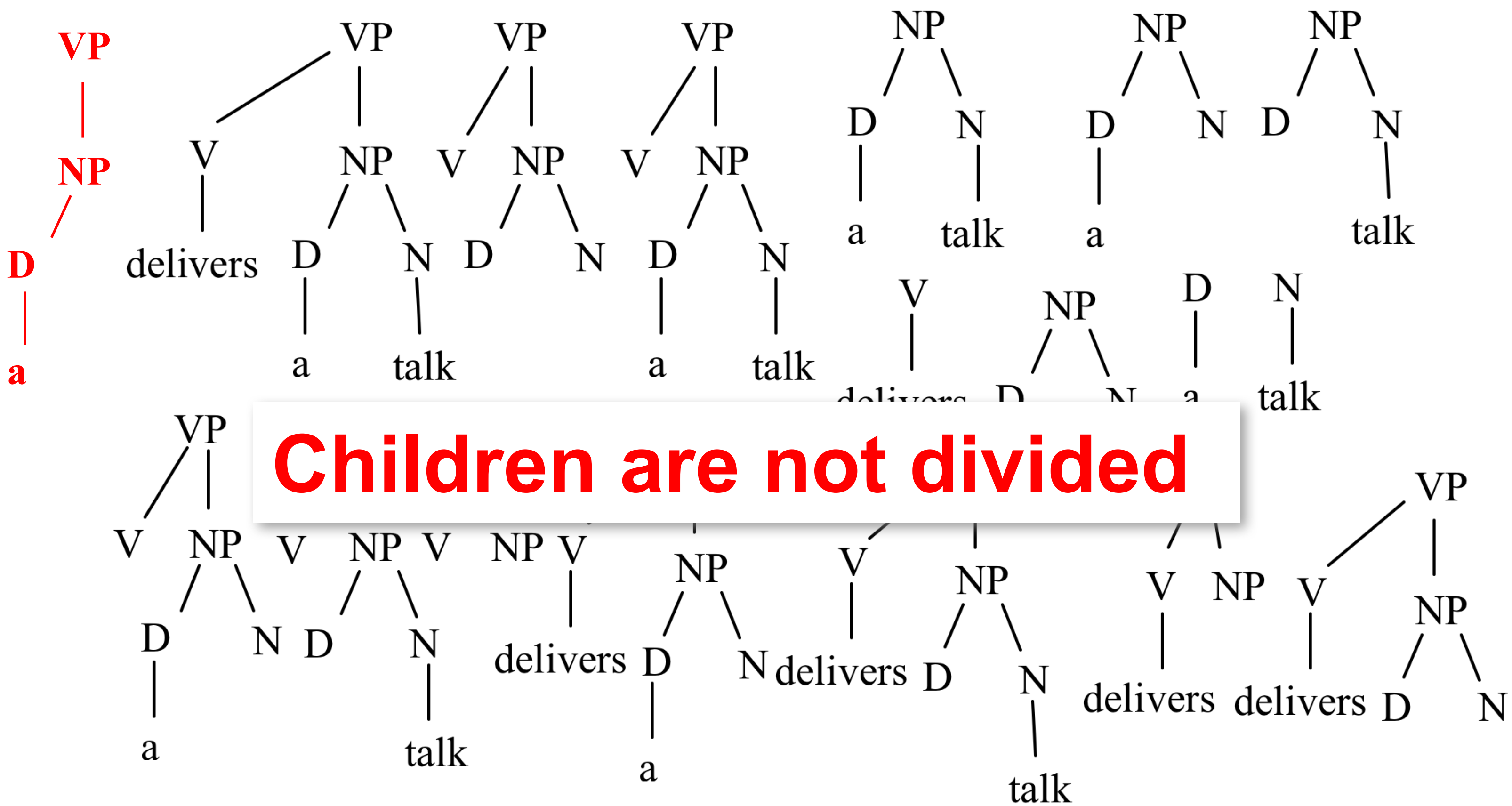
- “John delivers a talk in Rome”



The Syntactic Tree Kernel (STK)
[Collins and Duffy, 2002]

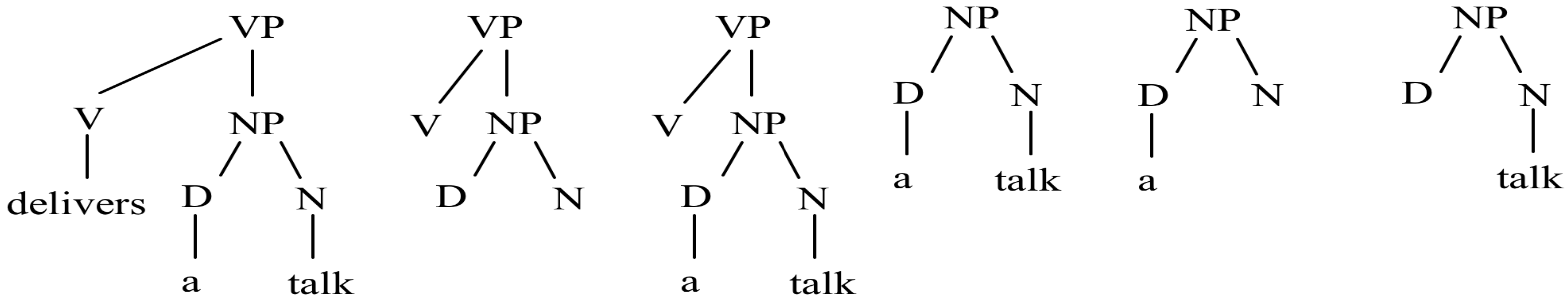


The overall fragment set

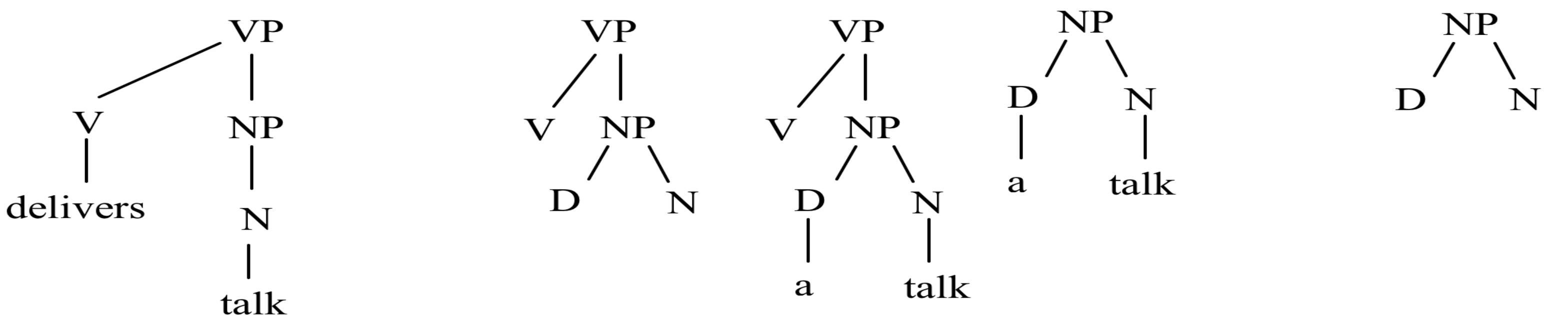


Explicit kernel space

$$\phi(T_x) = \vec{x} = (0,..,1,..,0,..,1,..,0,..,1,..,0,..,1,..,0,..,1,..,0,..,1,..,0)$$



$$\phi(T_z) = \vec{z} = (1,..,0,..,0,..,1,..,0,..,1,..,0,..,1,..,0,..,0,..,1,..,0,..,0)$$



- $\vec{x} \cdot \vec{z}$ counts the number of common substructures

Efficient evaluation of the scalar product

$$\vec{x} \cdot \vec{z} = \phi(T_x) \cdot \phi(T_z) = K(T_x, T_z) =$$

$$= \sum_{n_x \in T_x} \sum_{n_z \in T_z} \Delta(n_x, n_z)$$

Efficient evaluation of the scalar product

$$\vec{x} \cdot \vec{z} = \phi(T_x) \cdot \phi(T_z) = K(T_x, T_z) =$$

$$= \sum_{n_x \in T_x} \sum_{n_z \in T_z} \Delta(n_x, n_z)$$

- [Collins and Duffy, ACL 2002] evaluate Δ in $O(n^2)$:

$$\Delta(n_x, n_z) = 0, \quad \text{if the productions are different else}$$

$$\Delta(n_x, n_z) = 1, \quad \text{if pre-terminals else}$$

$$\Delta(n_x, n_z) = \prod_{j=1}^{nc(n_x)} (1 + \Delta(ch(n_x, j), ch(n_z, j)))$$

Other Adjustments

- Decay factor

$$\Delta(n_x, n_z) = \lambda, \quad \text{if pre - terminals else}$$

$$\Delta(n_x, n_z) = \lambda \prod_{j=1}^{nc(n_x)} (1 + \Delta(ch(n_x, j), ch(n_z, j)))$$

- Normalization

$$K'(T_x, T_z) = \frac{K(T_x, T_z)}{\sqrt{K(T_x, T_x) \times K(T_z, T_z)}}$$

Observations

- We can order the production rules used in T_x and T_z, at loading time
- At learning time we can evaluate NP in

$|T_x|+|T_z|$ *running time* [Moschitti, EACL 2006]

- If T_x and T_z are generated by only one production rule $\Rightarrow$ $O(|T_x|\times|T_z|)$…***Very Unlikely!!!!***

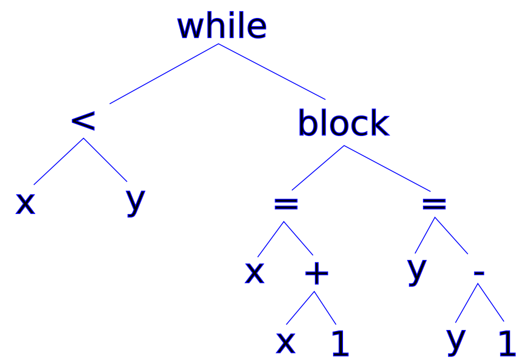
Trees can also be program derivation trees

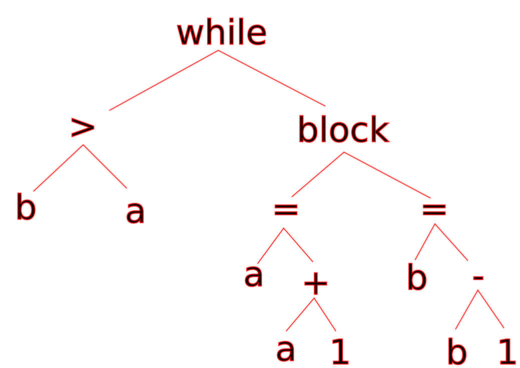
CODE

```
while (x < y) {
   x = x + 1
   y = y - 1
}
```

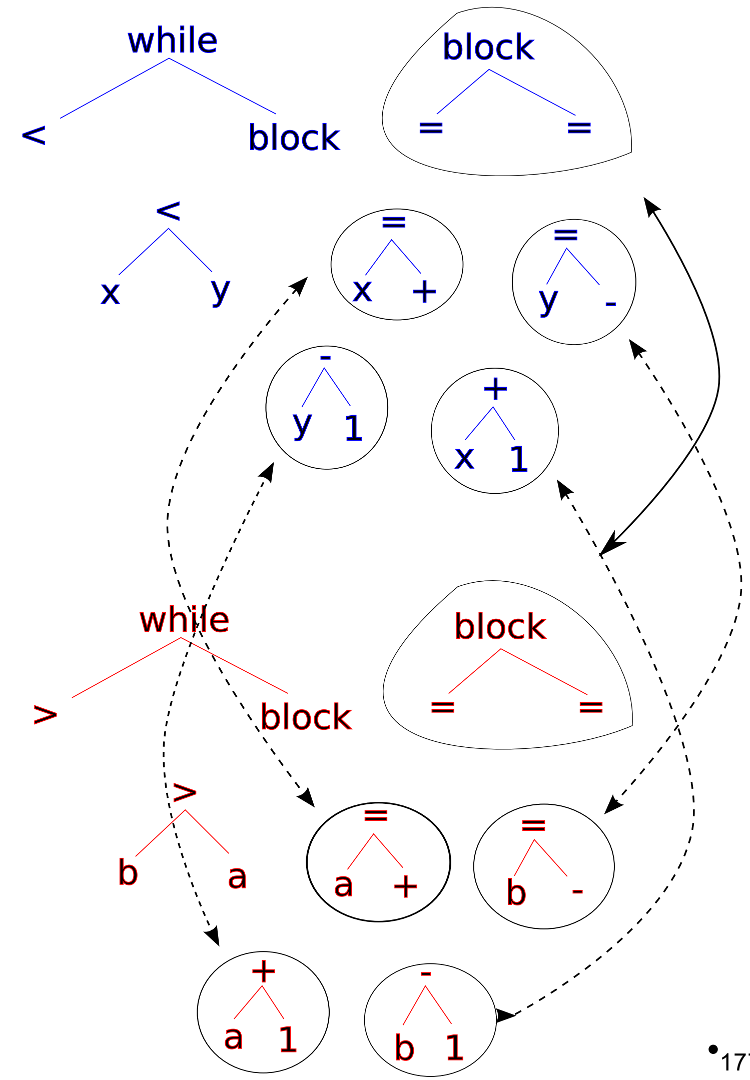
```
while (b > a) {
   a = a + 1
   b = b - 1
}
```

AST



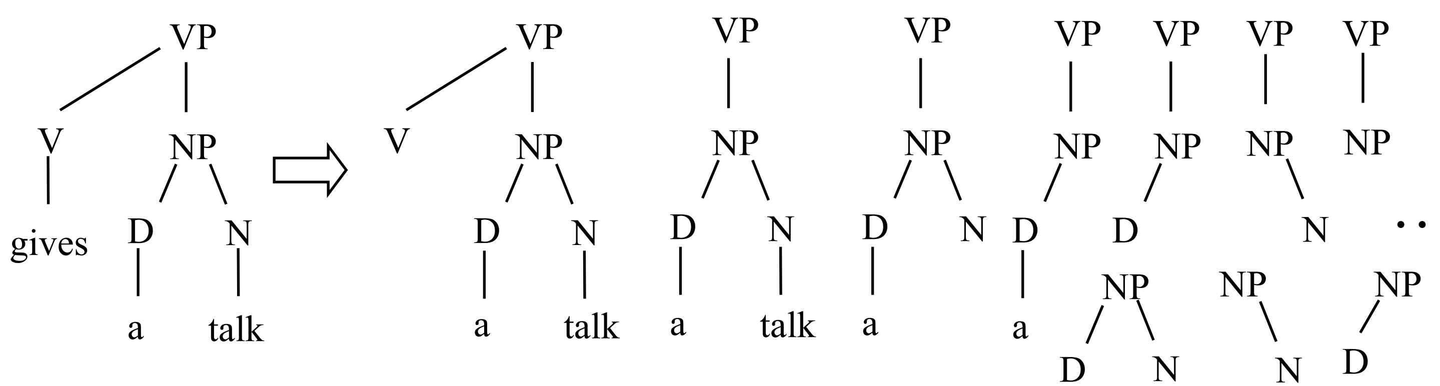


AST KERNEL

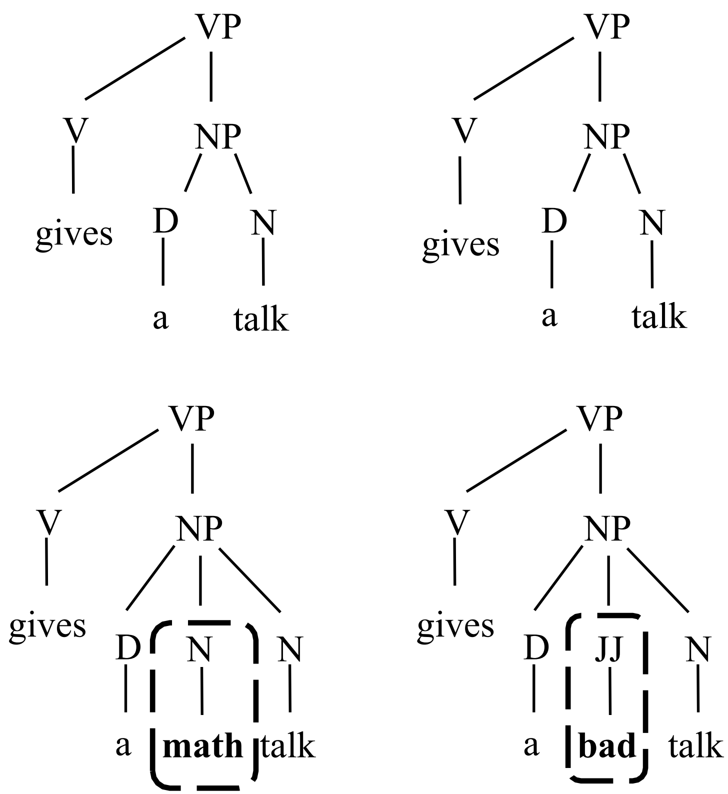


Labeled Ordered Tree Kernel

- STK satisfies the constraint “remove 0 or all children at a time”.
- If we relax such constraint we get more general substructures [Kashima and Koyanagi, 2002]



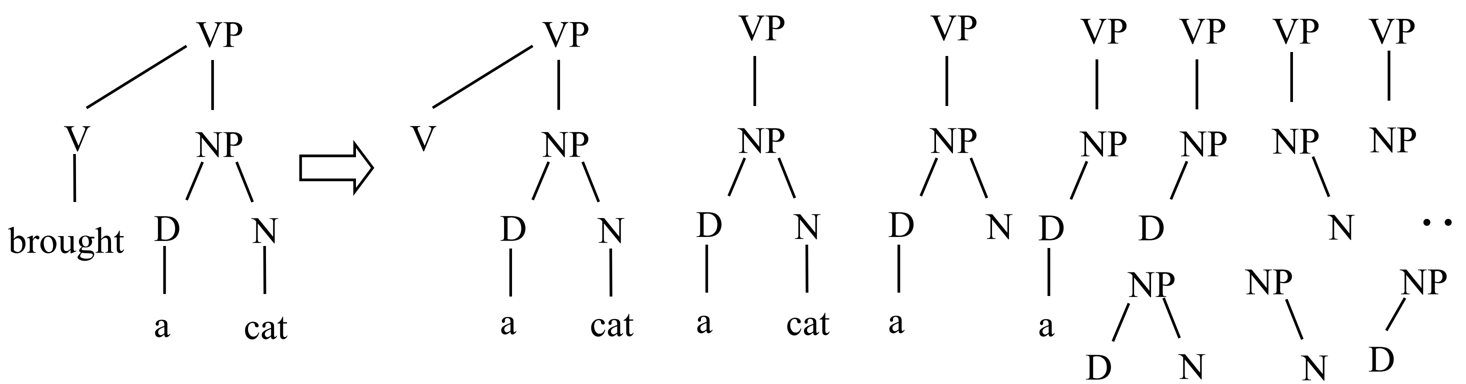
Weighting Problems



- Both matched pairs give the same contribution
- Gap based weighting is needed
- A novel efficient evaluation has to be defined

Partial Tree Kernel (PTK)
[Moschitti, ECML 2006]

- STK + String Kernel with weighted gaps on nodes' children



Partial Tree Kernel - Definition

- if the node labels of n_1 and n_2 are different then $\Delta(n_1, n_2) = 0$;
- else
$$\Delta(n_1, n_2) = 1 + \sum_{\vec{J}_1, \vec{J}_2, l(\vec{J}_1) = l(\vec{J}_2)} \prod_{i=1}^{l(\vec{J}_1)} \Delta(c_{n_1}[\vec{J}_{1i}], c_{n_2}[\vec{J}_{2i}])$$

- By adding two decay factors we obtain:

$$\mu\left(\lambda^2 + \sum_{\vec{J}_1, \vec{J}_2, l(\vec{J}_1) = l(\vec{J}_2)} \lambda^{d(\vec{J}_1) + d(\vec{J}_2)} \prod_{i=1}^{l(\vec{J}_1)} \Delta(c_{n_1}[\vec{J}_{1i}], c_{n_2}[\vec{J}_{2i}])\right)$$

Efficient Evaluation (1)

- In [Taylor and Cristianini, 2004 book], sequence kernels with weighted gaps are factorized with respect to different subsequence sizes
- We treat children as sequences and apply the same theory

$$\Delta(n_1, n_2) = \mu\Big(\lambda^2 + \sum_{p=1}^{lm} \Delta_p(c_{n_1}, c_{n_2})\Big)$$

Given the two child sequences $s_1 a = c_{n_1}$ and $s_2 b = c_{n_2}$ (a and b are the last children), $\Delta_p(s_1 a, s_2 b) =$

$$\Delta(a,b) \times \underbrace{\boxed{\sum_{i=1}^{|s_1|}\sum_{r=1}^{|s_2|} \lambda^{|s_1|-i+|s_2|-r} \times \Delta_{p-1}(s_1[1:i], s_2[1:r])}}_{D_p}$$

Efficient Evaluation (2)

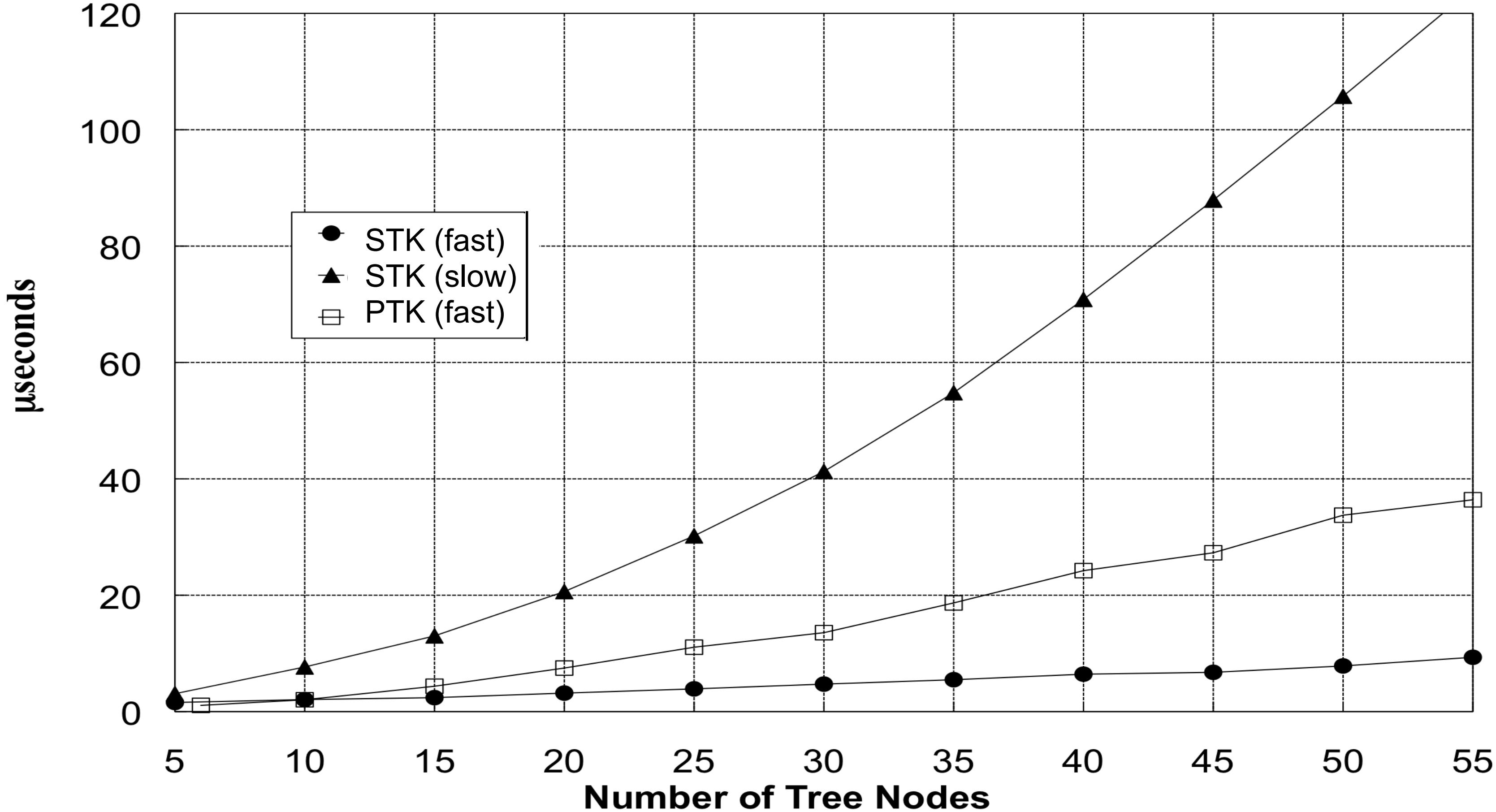
$$\Delta_p(s_1 a, s_2 b) = \begin{cases} \Delta(a,b) D_p(|s_1|, |s_2|) & \text{if } a = b; \\ 0 & otherwise. \end{cases}$$

Note that D_p satisfies the recursive relation:

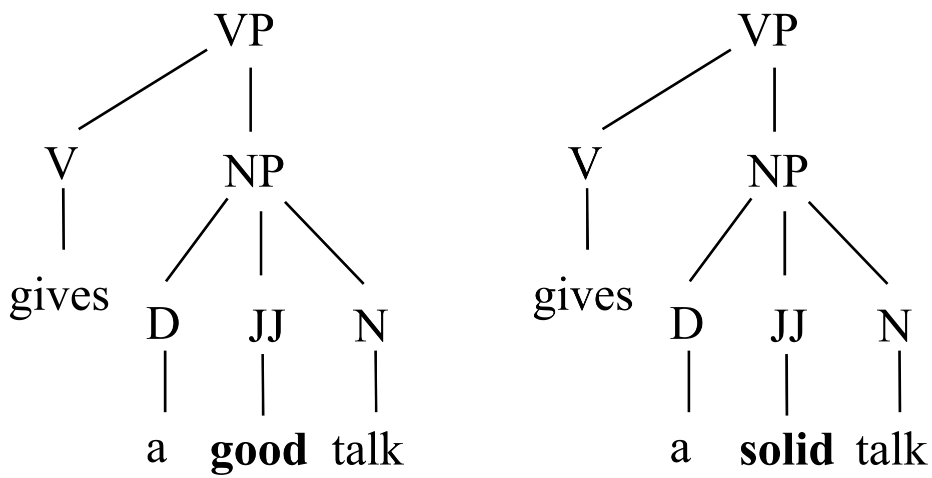
$$\begin{aligned} D_p(k,l) &= \Delta_{p-1}(s_1[1:k], s_2[1:l]) + \lambda D_p(k, l-1) \\ &\quad + \lambda D_p(k-1, l) + \lambda^2 D_p(k-1, l-1). \end{aligned}$$

- The complexity of finding the subsequences is $O(p|s_1||s_2|)$
- Therefore the overall complexity is $O(p\rho^2|N_{T_1}||N_{T_2}|)$ where ρ is the maximum branching factor ($p = \rho$)

Running Time of Tree Kernel Functions



Syntactic/Semantic Tree Kernels (SSTK)
[Bloehdorn & Moschitti, ECIR 2007 & CIKM 2007]



- Similarity between the fragment leaves
  - Tree kernel + Lexical Similarity Kernel

Equations of SSTK

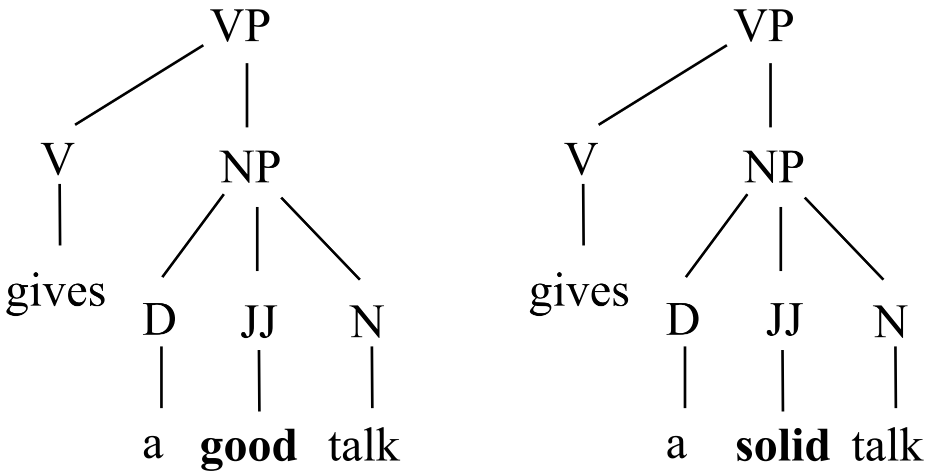
Definition 4 (Tree Fragment Similarity Kernel). *For two tree fragments $f_1, f_2 \in \mathcal{F}$, we define the Tree Fragment Similarity Kernel as*[6]*:*

$$\kappa_{\mathcal{F}}(f_1, f_2) = comp(f_1, f_2) \prod_{t=1}^{nt(f_1)} \kappa_S(f_1(t), f_2(t))$$

$$\kappa_T(T_1, T_2) = \sum_{n_1 \in N_{T_1}} \sum_{n_2 \in N_{T_2}} \Delta(n_1, n_2)$$

where $\Delta(n_1, n_2) = \sum_{i=1}^{|\mathcal{F}|} \sum_{j=1}^{|\mathcal{F}|} I_i(n_1) I_j(n_2) \kappa_{\mathcal{F}}(f_i, f_j)$.

Example of an SSTK evaluation



K_S(gives,gives)*K_S(a,a)*

K_S(good,solid)*K_S(talk,talk)

= 1 * 1 * 0.5 * 1 = 0.5

$$\kappa_T(T_1, T_2) = \sum_{n_1 \in N_{T_1}} \sum_{n_2 \in N_{T_2}} \Delta(n_1, n_2)$$

$$where\ \Delta(n_1, n_2) = \sum_{i=1}^{|\mathcal{F}|} \sum_{j=1}^{|\mathcal{F}|} I_i(n_1) I_j(n_2) \kappa_{\mathcal{F}}(f_i, f_j).$$

Delta Evaluation is very simple

0. if n_1 and n_2 are pre-terminals and $label(n_1) = label(n_2)$ then $\Delta(n_1, n_2) = \lambda \kappa_{\mathcal{S}}(ch^1_{n_1}, ch^1_{n_2})$,

1. if the productions at n_1 and n_2 are different then $\Delta(n_1, n_2) = 0$;

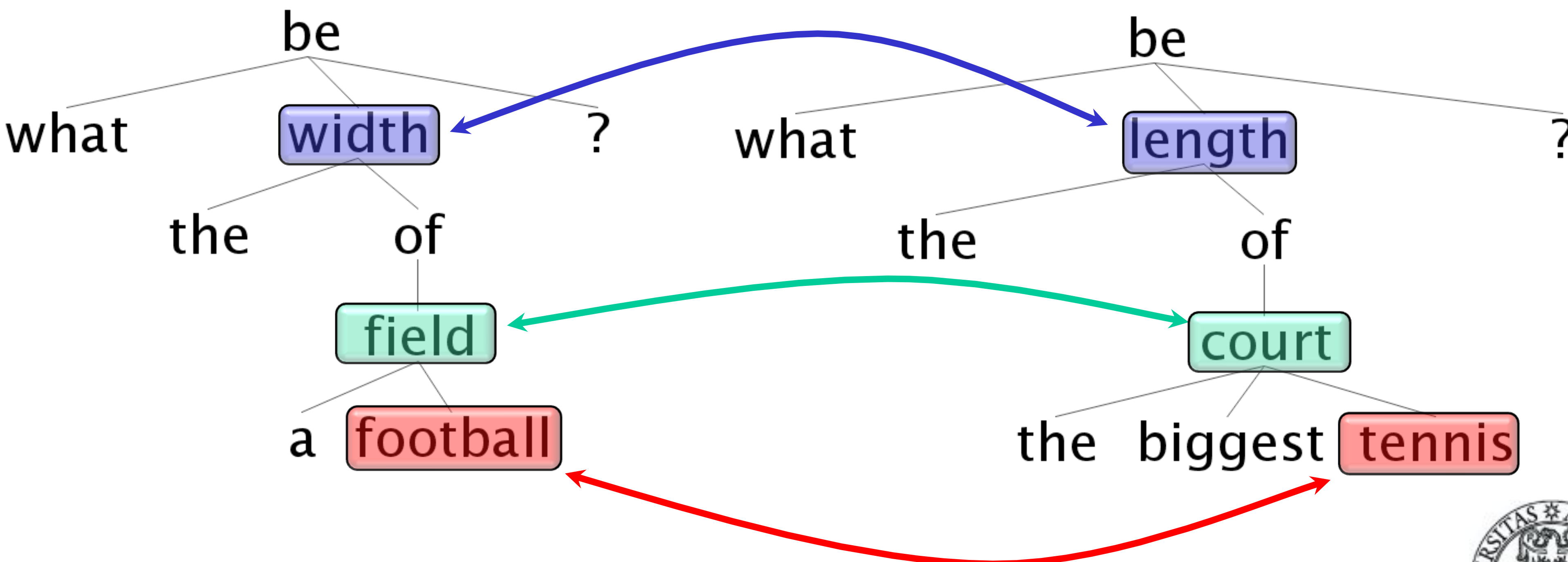
2. $\Delta(n_1, n_2) = \lambda$,
3. $\Delta(n_1, n_2) = \lambda \prod_{j=1}^{nc(n_1)} (1 + \Delta(ch^j_{n_1}, ch^j_{n_2}))$.

Smoothed Partial Tree Kernels
[Moschitti, EACL 2009; Croce et al., 2011]

- Same idea of Syntactic Semantic Tree Kernel but the similarity is extended to any node of the tree
- The tree fragments are those generated by PTK
- Basically it extends PTK with similarities

Examples of Dependency Trees

- What is the width of a football field?
- What is the length of the biggest tennis court?



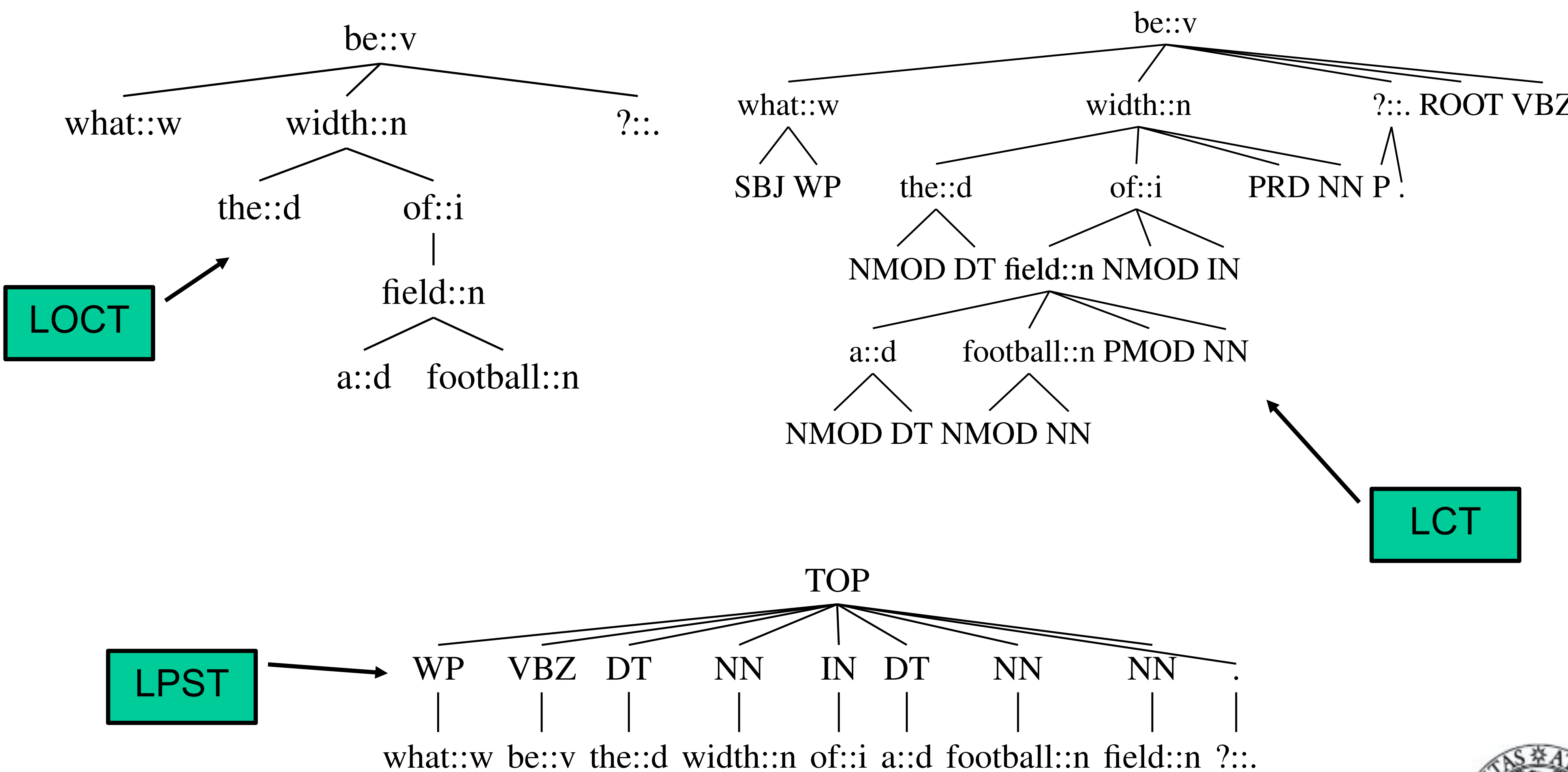
Equation of SPTK

If n_1 and n_2 are leaves then $\Delta_\sigma(n_1, n_2) = \boxed{\mu\lambda\sigma(n_1, n_2)}$

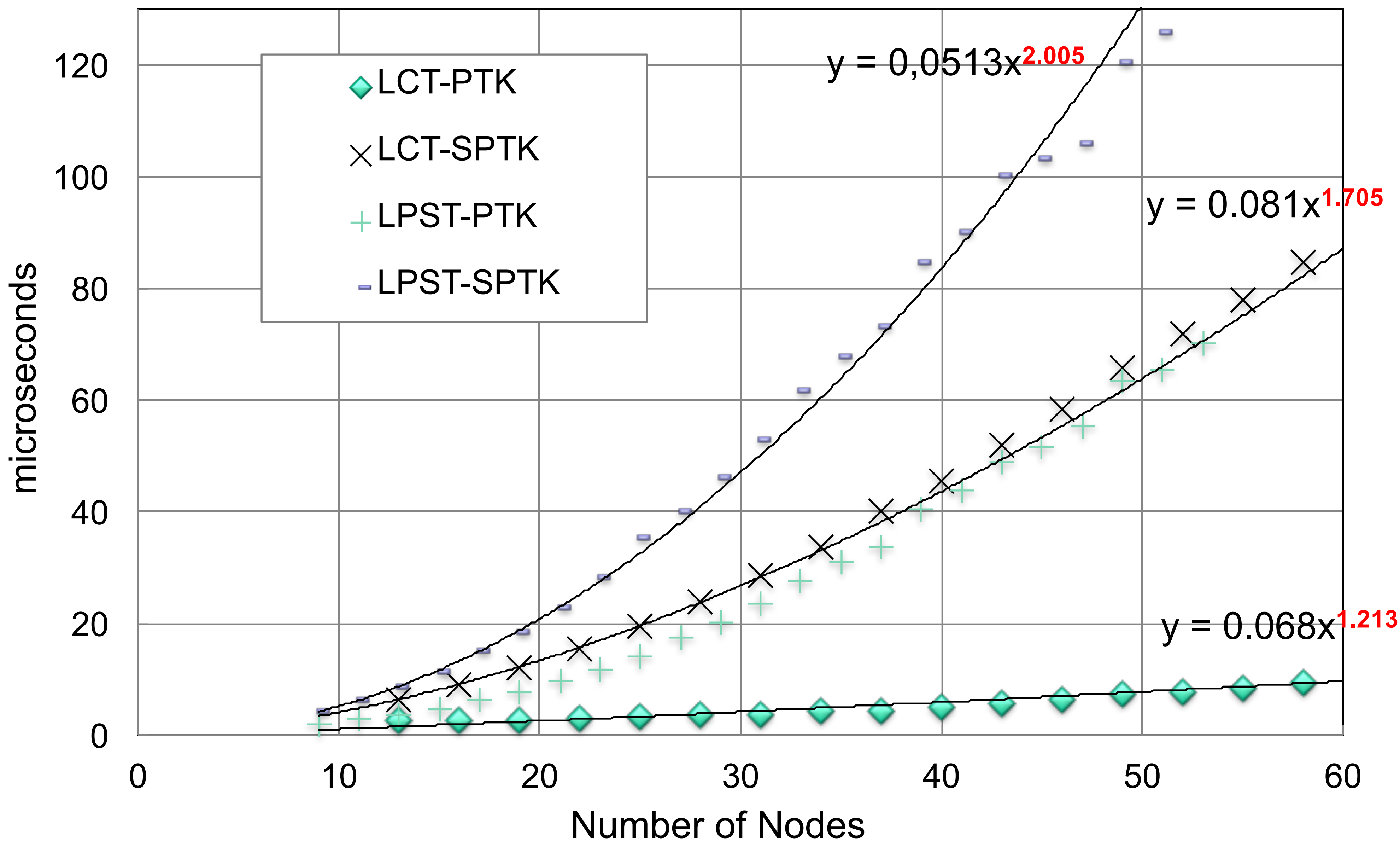
else

$$\Delta_\sigma(n_1,n_2) = \underbrace{\boxed{\mu\sigma(n_1,n_2)}}_{\textbf{Lexical Similarity}} \times \underbrace{\boxed{\left(\lambda^2 + \sum_{\vec{I}_1,\vec{I}_2,l(\vec{I}_1)=l(\vec{I}_2)} \lambda^{d(\vec{I}_1)+d(\vec{I}_2)} \prod_{j=1}^{l(\vec{I}_1)} \Delta_\sigma(c_{n_1}(\vec{I}_{1j}), c_{n_2}(\vec{I}_{2j}))\right)}}_{\textbf{PTK}}$$

Different versions of Computational Dependency Trees for PTK/SPTK



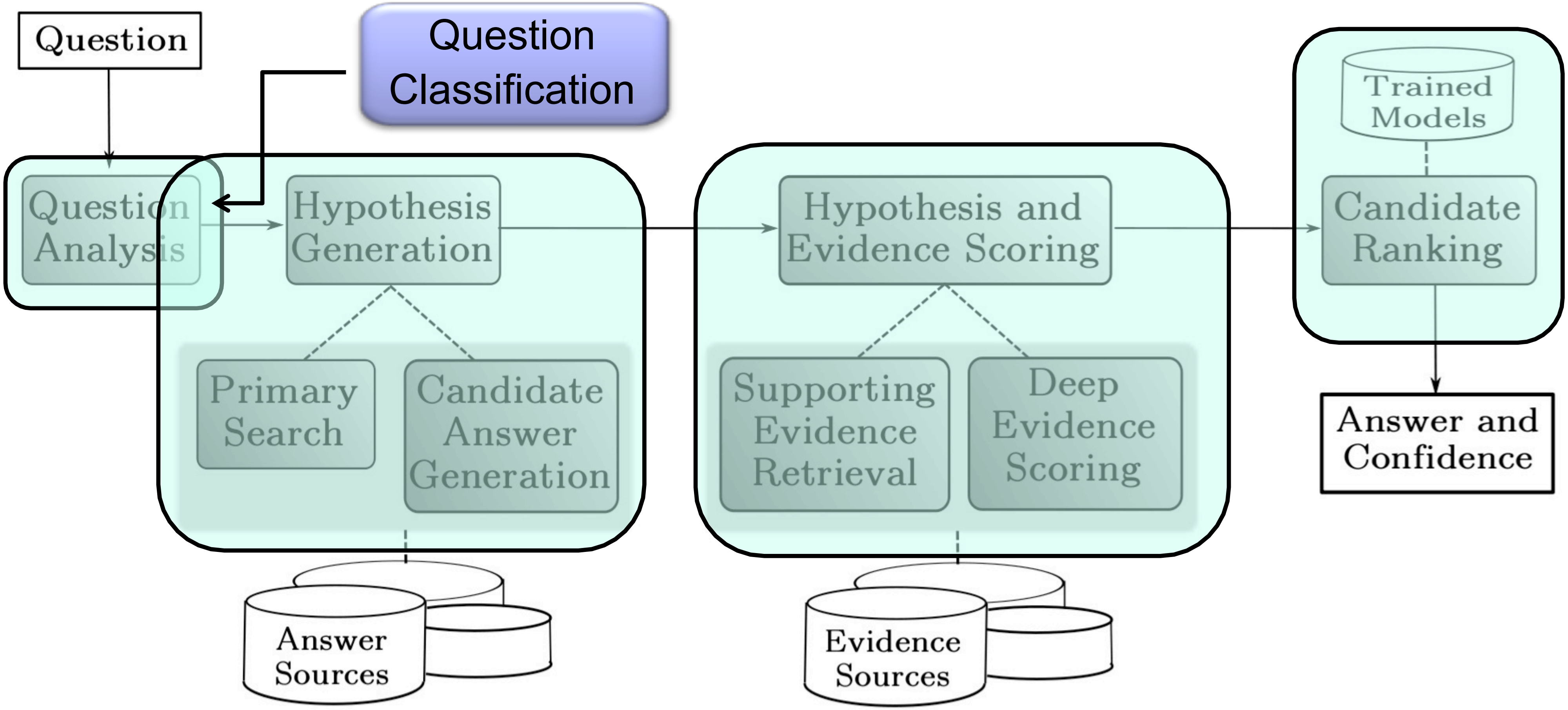
Tree Kernel Efficiency



Outline: Part I – Classification with Kernels

- Classification with Kernels (15 min)
  - Question Classification (QC) using constituency, dependency and semantic structures
  - Question Classification (QC) in Jeopardy!
  - Relation Extraction with kernels
  - Kernel-Based Coreference Resolution

IBM Watson (simplified) Pipeline



Question Classification

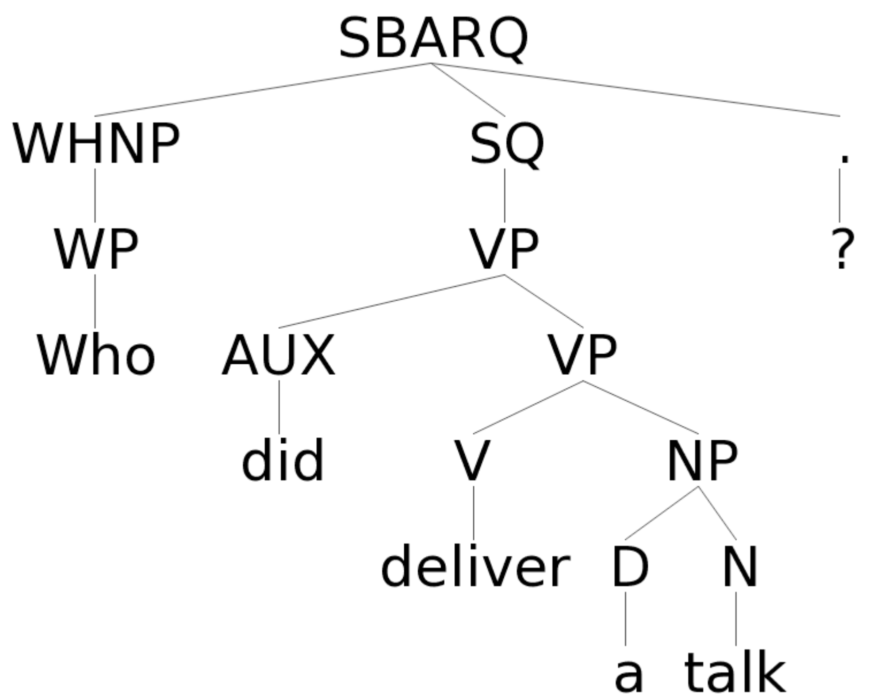
- **Definition**: What does HTML stand for?
- **Description**: What's the final line in the Edgar Allan Poe poem "The Raven"?
- **Entity**: What foods can cause allergic reaction in people?
- **Human**: Who won the Nobel Peace Prize in 1992?
- **Location**: Where is the Statue of Liberty?
- **Manner**: How did Bob Marley die?
- **Numeric**: When was Martin Luther King Jr. born?
- **Organization**: What company makes Bentley cars?

Question Classifier based on Tree Kernels

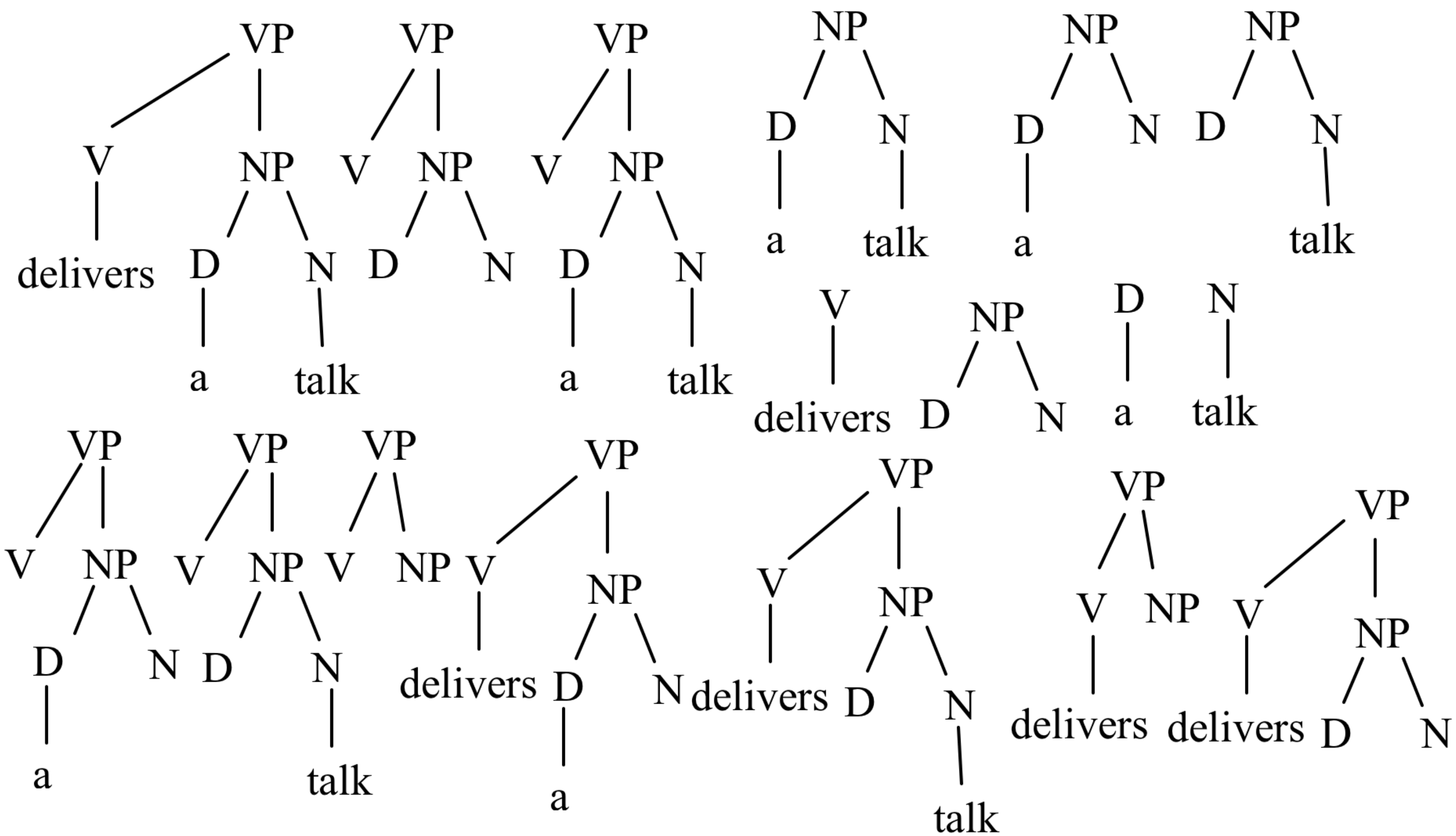
- Question dataset (http://l2r.cs.uiuc.edu/~cogcomp/Data/QA/QC/) [Lin and Roth, 2005])
  - Distributed on 6 categories: Abbreviations, Descriptions, Entity, Human, Location, and Numeric.
- Fixed split 5500 training and 500 test questions
- Using the whole question parse trees
  - Constituent parsing
  - Example

  "**Who did deliver a talk?**"

Syntactic Parse Trees (PT)

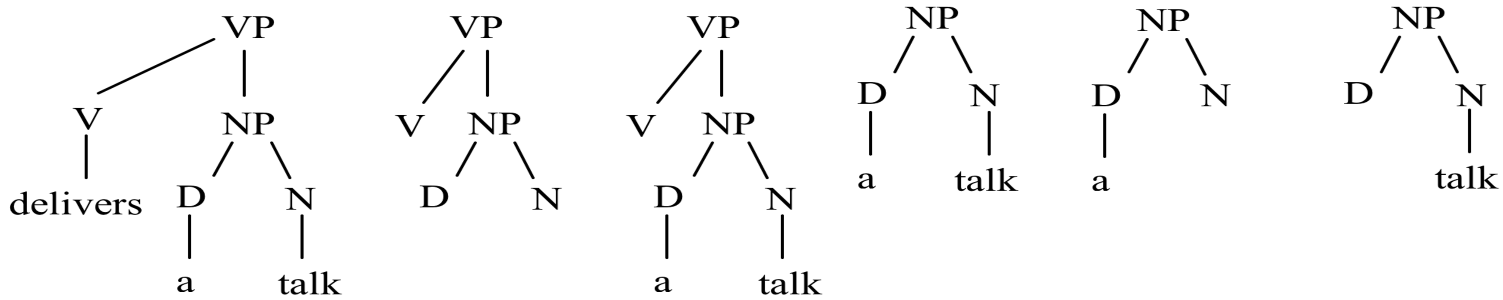


Some fragments from the VP subtree

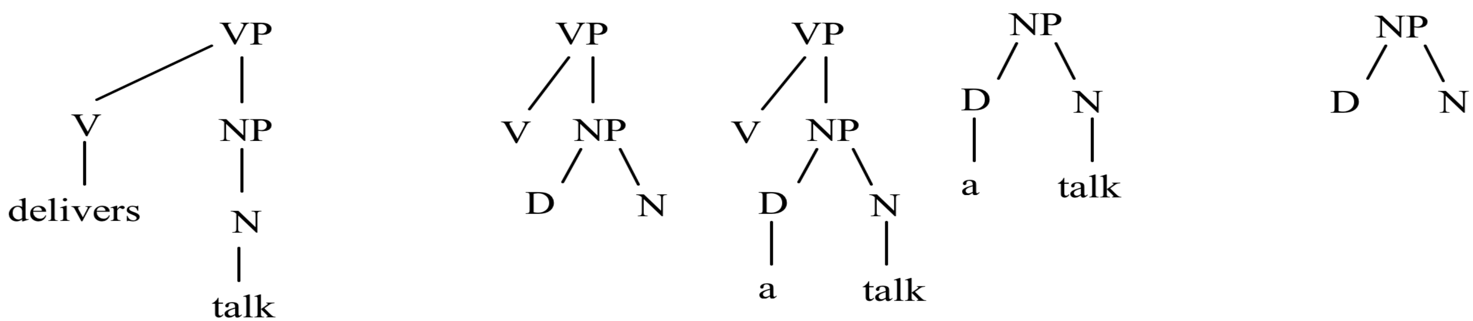


Explicit kernel space

$$\phi(T_x) = \vec{x} = (0,..,1,..,0,..,1,..,0,..,1,..,0,..,1,..,0,..,1,..,0,..,1,..,0)$$



$$\phi(T_z) = \vec{z} = (1,..,0,..,0,..,1,..,0,..,1,..,0,..,1,..,0,..,0,..,1,..,0,..,0)$$



- $\vec{x} \cdot \vec{z}$ counts the number of common substructures

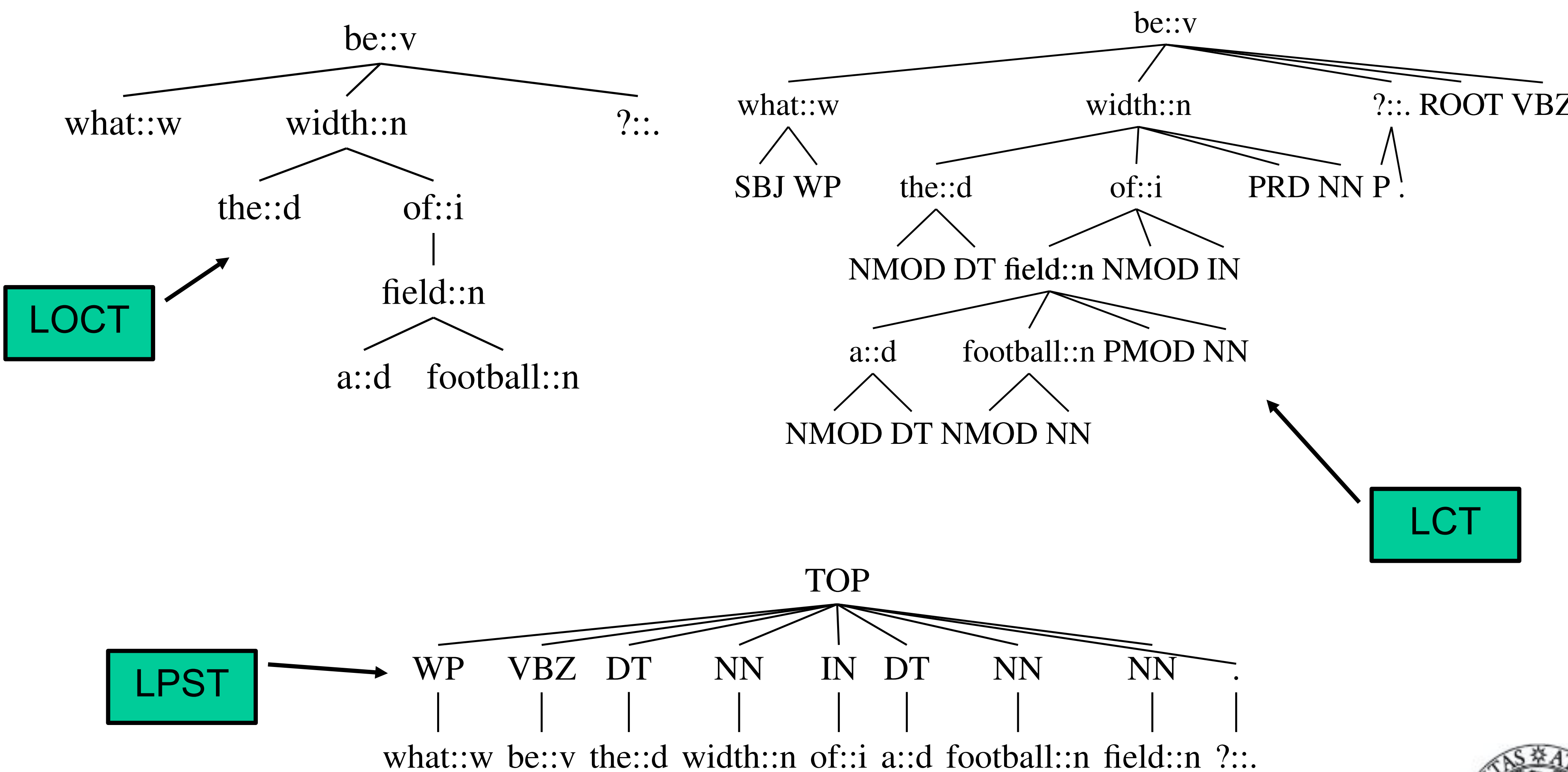
Question Classification with SSTK
[Blohedorn&Moschitti, CIKM2007]

Syntactic Tree Kernel
(STK)

	Accuracy				
λ **parameter**	**0.4**	**0.05**	**0.01**	**0.005**	**0.001**
linear (bow)	0.905				
string matching	0.890	0.910	**0.914**	**0.914**	0.912
full	0.904	**0.924**	0.918	0.922	0.920
full-ic	0.908	**0.922**	0.916	0.918	0.918
path-1	0.906	**0.918**	0.912	**0.918**	0.916
path-2	0.896	0.914	0.914	**0.916**	**0.916**
lin	0.908	**0.924**	0.918	0.922	0.922
wup	0.908	**0.926**	0.918	0.922	0.922

Syntactic Tree Kernel
with similarities (SSTK)

Same Task with PTK, SPTK and Dependency Trees



State-of-the-art Results
[Croce et al., EMNLP 2011]

	STK	PTK	SPTK(LSA)
CT	91.20%	90.80%	91.00%
LOCT	-	89.20%	93.20%
LCT	-	90.80%	**94.80%**
LPST	-	89.40%	89.60%
BOW		88.80%	

Classification, Ranking, Regression and Multiclassification

The Ranking SVM
[Herbrich et al. 1999, 2000; Joachims et al. 2002]

- The aim is to classify instance pairs as correctly ranked or incorrectly ranked
  - This turns an ordinal regression problem back into a binary classification problem
- We want a ranking function *f* such that

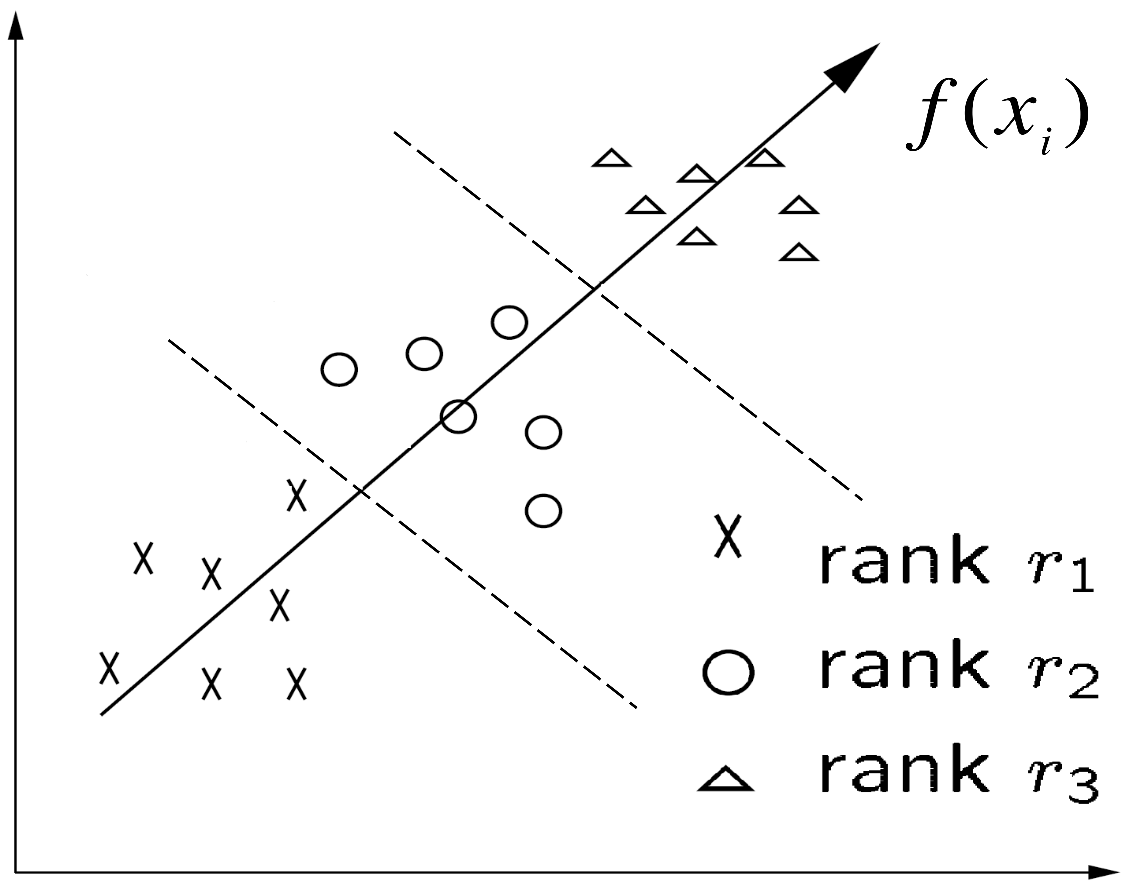
$$\boldsymbol{x}_i > \boldsymbol{x}_j \text{ iff } f(\boldsymbol{x}_i) > f(\boldsymbol{x}_j)$$

- … or at least one that tries to do this with minimal error
- Suppose that *f* is a linear function

$$f(\boldsymbol{x}_i) = \mathbf{w} \bullet \boldsymbol{x}_i$$

The Ranking SVM

- Ranking Model: $f(\boldsymbol{x}_i)$



The Ranking SVM

- Then (combining the two equations on the last slide):

$\boldsymbol{x}_i > \boldsymbol{x}_j$ iff $\mathbf{w}\bullet\boldsymbol{x}_i - \mathbf{w}\bullet\boldsymbol{x}_j > 0$

$\boldsymbol{x}_i > \boldsymbol{x}_j$ iff $\mathbf{w}\bullet(\boldsymbol{x}_i - \boldsymbol{x}_j) > 0$

- Let us then create a new instance space from such pairs: $z_k = x_i - x_k$

$$y_k = +1, -1 \text{ as } \boldsymbol{x}_i \geq, < \boldsymbol{x}_k$$

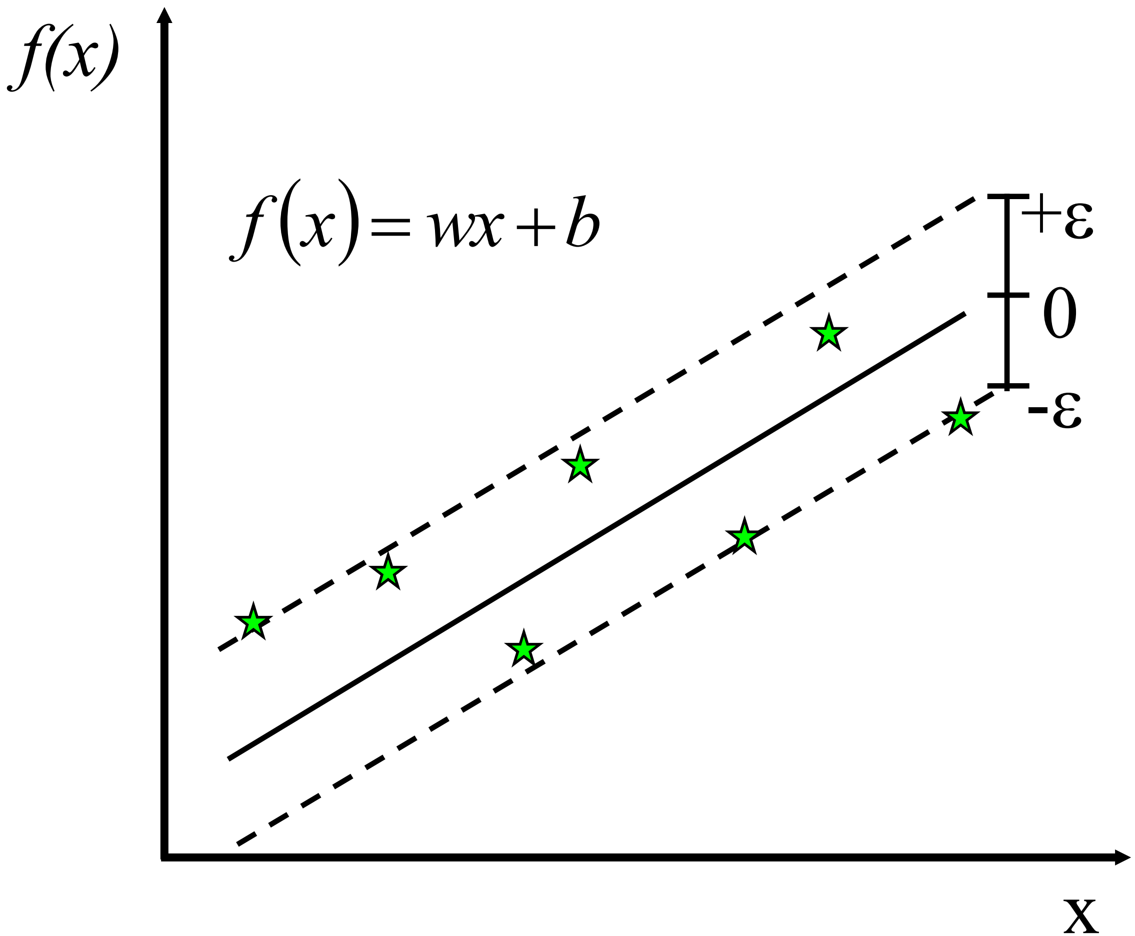
Support Vector Ranking

$$\begin{cases} min \quad \frac{1}{2}||\vec{w}|| + C\sum_{i=1}^{m}\xi_i^2 \\ y_k(\vec{w}\cdot(\vec{x_i}-\vec{x_j})+b) \geq 1-\xi_k, \quad \forall i,j=1,..,m \\ \xi_k \geq 0, \quad k=1,..,m^2 \end{cases}$$

$y_k = 1$ if $rank(\vec{x_i}) > rank(\vec{x_j})$, -1 otherwise, where $k = i \times m + j$

- Given two examples we build one example (x_i , x_j)

Support Vector Regression (SVR)



Solution:

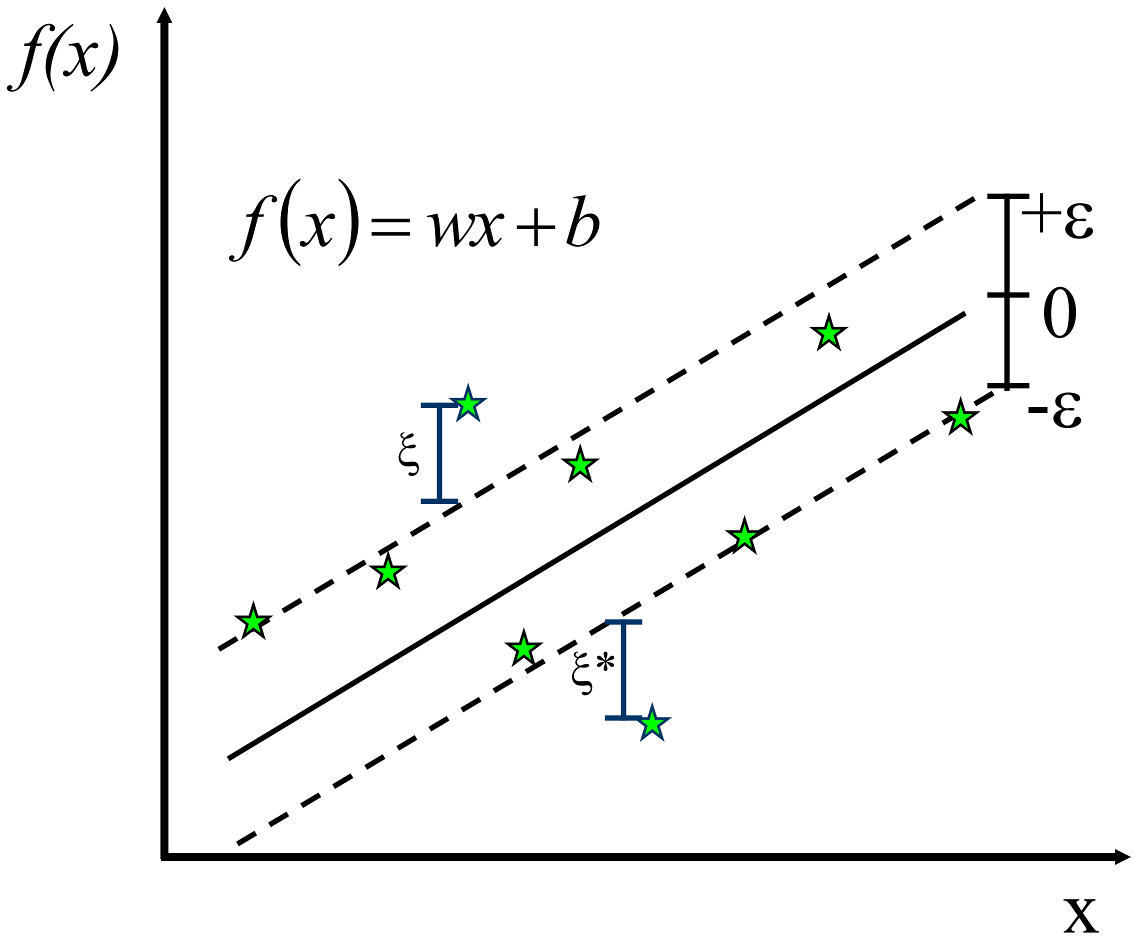
$$Min \frac{1}{2} w^T w$$

Constraints:

$$y_i - w^T x_i - b \leq \varepsilon$$

$$w^T x_i + b - y_i \leq \varepsilon$$

Support Vector Regression (SVR)



Minimise:

$$\frac{1}{2} w^T w + \boxed{C \sum_{i=1}^{N} \left(\xi_i + \xi_i^* \right)}$$

Constraints:

$$y_i - w^T x_i - b \le \varepsilon \boxed{+ \xi_i}$$

$$w^T x_i + b - y_i \le \varepsilon \boxed{+ \xi_i^*}$$

$$\boxed{\xi_i, \xi_i^* \ge 0}$$

Support Vector Regression

$$\min_{\mathbf{w},b,\xi,\xi^*} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i^*)$$

$$\begin{aligned} \text{s.t. } & y_i - \mathbf{w}^\top \mathbf{x}_i - b \le \epsilon + \xi_i,\ \ \xi_i \ge 0 \quad \forall 1 \le i \le n; \\ & \mathbf{w}^\top \mathbf{x}_i + b - y_i \le \epsilon + \xi_i^*,\ \ \xi_i^* \ge 0 \quad \forall 1 \le i \le n. \end{aligned}$$

- y_i is not -1 or 1 anymore, now it is a value
- ε is the tollerance of our function value

From Binary to Multiclass classifiers

- Three different approaches:
- **ONE-vs-ALL (OVA)**
  - Given the example sets, {E1, E2, E3, …} for the categories: {C1, C2, C3,…} the binary classifiers: {b1, b2, b3,…} are built.
  - For b1, E1 is the set of positives and E2∪E3 ∪… is the set of negatives, and so on
  - For testing: given a classification instance x, the category is the one associated with the maximum margin among all binary classifiers

From Binary to Multiclass classifiers

- **ALL-vs-ALL (AVA)**
  - Given the examples: {E1, E2, E3, …} for the categories {C1, C2, C3,…}
    - build the binary classifiers:

      {b1_2, b1_3,…, b1_n, b2_3, b2_4,…, b2_n,…,bn-1_n}
    - by learning on E1 (positives) and E2 (negatives), on E1 (positives) and E3 (negatives) and so on…
  - For testing: given an example x,
    - all the votes of all classifiers are collected
    - where $b_{E1E2} = 1$ means a vote for C1 and $b_{E1E2} = -1$ is a vote for C2
  - Select the category that gets more votes

Natural Language Processing and Information Retrieval

Structured Output

Alessandro Moschitti

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

Simple Structured Output

- We have seen methods for: binary Classifier or multiclassifier single label
- Multiclass-Multilabel is a structured output, i.e. a label subset is output

From Binary to Multiclass classifiers

- Three different approaches:
- **ONE-vs-ALL (OVA)**
  - Given the example sets, {E1, E2, E3, …} for the categories: {C1, C2, C3,…} the binary classifiers: {b1, b2, b3,…} are built.
  - For b1, E1 is the set of positives and E2∪E3 ∪… is the set of negatives, and so on
  - <u>For testing</u>: given a classification instance x, the category is the one associated with the maximum margin among all binary classifiers

From Binary to Multiclass classifiers

- **ALL-vs-ALL (AVA)**
  - Given the examples: {E1, E2, E3, …} for the categories {C1, C2, C3,…}
    - build the binary classifiers:

      {b1_2, b1_3,…, b1_n, b2_3, b2_4,…, b2_n,…,bn-1_n}
    - by learning on E1 (positives) and E2 (negatives), on E1 (positives) and E3 (negatives) and so on…
  - For testing: given an example x,
    - all the votes of all classifiers are collected
    - where $b_{E1E2} = 1$ means a vote for C1 and $b_{E1E2} = -1$ is a vote for C2
  - Select the category that gets more votes

From Binary to Multiclass classifiers

- **Error Correcting Output Codes** (ECOC)
  - The training set is partitioned according to binary sequences (codes) associated with category sets.
    - For example, 10101 indicates that the set of examples of C1,C3 and C5 are used to train the C_{10101} classifier.
    - The data of the other categories, i.e. C2 and C4 will be negative examples
  - In testing: the code-classifiers are used to decode one the original class, e.g.

    $C_{10101} = 1$ and $C_{11010} = 1$ indicates that the instance belongs to C1

That is, the only one consistent with the codes

Designing Global Classifiers

- Each class has a parameter vector (w_k, b_k)
- x is assigned to class k iff

$$w_k^\top x + b_k \geq \max_j w_j^\top x + b_j$$

- For simplicity set b_k=0
(add a dimension and include it in w_k)
- The goal (given separable data) is to choose w_k s.t.

$$\forall (x^i, y^i), \quad {w_{y^i}}^\top x^i \geq \max_j w_j^\top x^i$$

Multi-class SVM

Primal problem: QP

$$\min_{w_1,\ldots,w_K} \quad \frac{1}{2}\|(w_1,\ldots,w_K)\|^2 + C\sum_{ik}\xi_{ik}$$

$$\text{s.t.} \quad \forall (i,k), \quad {w_{y^i}}^\top x^i - w_k^\top x^i \geq \mathbf{1}\{k \neq y^i\} - \xi_{ik}$$

Structured Output Model

- Main idea: define scoring function which **decomposes** as sum of features scores k on "**parts**" p:

$$score(\mathbf{x}, \mathbf{y}, \mathbf{w}) = \mathbf{w}^\top \Phi(\mathbf{x}, \mathbf{y}) = \sum_{k,p} w_k^\top \phi_k(\mathbf{x}_p, \mathbf{y}_p)$$

- Label examples by looking for max score:

$$prediction(\mathbf{x}, \mathbf{w}) = \operatorname*{arg\,max}_{\mathbf{y} \in \mathcal{Y}(\mathbf{x})} score(\mathbf{x}, \mathbf{y}, \mathbf{w})$$

space of feasible outputs

- Parts = nodes, edges, etc.

Structured Perceptron

Inputs: Training set (x_i, y_i) for $i = 1 \ldots n$

Initialization: $\mathbf{W} = 0$

Define: $F(x) = \mathrm{argmax}_{y \in \mathbf{GEN}(x)} \mathbf{\Phi}(x, y) \cdot \mathbf{W}$

Algorithm: For $t = 1 \ldots T, i = 1 \ldots n$

$z_i = F(x_i)$

If $(z_i \neq y_i)$ $\quad \mathbf{W} = \mathbf{W} + \mathbf{\Phi}(x_i, y_i) - \mathbf{\Phi}(x_i, z_i)$

Output: Parameters $\mathbf{W}$

(Averaged) Perceptron

For each datapoint $\mathbf{x}^i$

Predict: $\color{blue}{\hat{\mathbf{y}}_i} = \underset{\mathbf{y} \in \mathcal{Y}}{\arg\max}\ \mathbf{w}_t^\top \Phi(\mathbf{x}^i, \mathbf{y})$

Update: $\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha \underbrace{\left(\Phi(\mathbf{x}, \mathbf{y}^i) - \Phi(\mathbf{x}^i, \hat{\mathbf{y}}_i)\right)}_{\text{update if } \hat{\mathbf{y}}_i \neq \mathbf{y}^i}$

Averaged perceptron: $\bar{\mathbf{w}} = \frac{1}{T} \sum_{t=1}^{T} \mathbf{w}_t$

Example: multiclass setting

Predict: $\hat{y}_i = \arg\max_y \; w_y^\top x^i$

Update: if $\hat{y}_i \neq y^i$ then

$$w_{y^i,t+1} = w_{y^i,t} + \alpha x^i$$
$$w_{\hat{y}_i,t+1} = w_{\hat{y}_i,t} - \alpha x^i$$

Feature encoding:

$$\begin{aligned}
\Phi(\mathbf{x}^i, y=1)^\top &= [\mathbf{x}^{i\top} \; 0 \ldots 0] \\
\Phi(\mathbf{x}^i, y=2)^\top &= [0 \; \mathbf{x}^{i\top} \ldots 0] \\
&\vdots \\
\Phi(\mathbf{x}^i, y=K)^\top &= [0 \; 0 \ldots \mathbf{x}^{i\top}] \\
\mathbf{w}^\top &= [w_1^\top \; w_2^\top \ldots w_K^\top]
\end{aligned}$$

Predict: $\hat{\mathbf{y}}_i = \arg\max_{\mathbf{y} \in \mathcal{Y}} \; \mathbf{w}_t^\top \Phi(\mathbf{x}^i, \mathbf{y})$

Update: $\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha \underbrace{\left(\Phi(\mathbf{x}, \mathbf{y}^i) - \Phi(\mathbf{x}^i, \hat{\mathbf{y}}_i)\right)}_{\text{update if } \hat{\mathbf{y}}_i \neq \mathbf{y}^i}$

Output of Ranked Example List

Support Vector Ranking

$$\begin{cases} min \quad \frac{1}{2}||\vec{w}|| + C\sum_{i=1}^{m}\xi_i^2 \\ y_k(\vec{w}\cdot(\vec{x_i}-\vec{x_j})+b) \geq 1-\xi_k, \quad \forall i,j=1,..,m \\ \xi_k \geq 0, \quad k=1,..,m^2 \end{cases}$$

y_k = 1 if $rank(\vec{x_i}) > rank(\vec{x_j})$, 0 otherwise, where $k = i \times m + j$

- Given two examples we build one example (x_i , x_j)

Concept Segmentation and Classification task

- Given a transcription, i.e. a sequence of words, chunk and label subsequences with concepts
- Air Travel Information System (ATIS)
  - Dialog systems answering user questions
  - Conceptually annotated dataset
  - Frames

An example of concept annotation in ATIS

- User request: ***list TWA flights from Boston to Philadelphia***

$$\underbrace{list}_{null}\ \underbrace{TWA}_{airline\_code}\ \underbrace{flights}_{null}\ \underbrace{from}_{null}\ \underbrace{Boston}_{fromloc.city}\ \underbrace{to}_{null}\ \underbrace{Philadelphia}_{toloc.city}$$

- The concepts are used to build rules for the dialog manager (e.g. actions for using the DB)
  - from location
  - to location
  - airline code

list flights from boston to Philadelphia		
FRAME:	FLIGHT	
	FROMLOC.CITY = boston	
	TOLOC.CITY = Philadelphia	

Our Approach
(Dinarelli, Moschitti, Riccardi, SLT 2008)

- Use of Finite State Transducer to generate word sequences and concepts
- Probability of each annotation

$\Rightarrow$ *m* best hypothesis can be generated

- Idea: use a discriminative model to choose the best one
  - Re-ranking and selecting the top one

Experiments

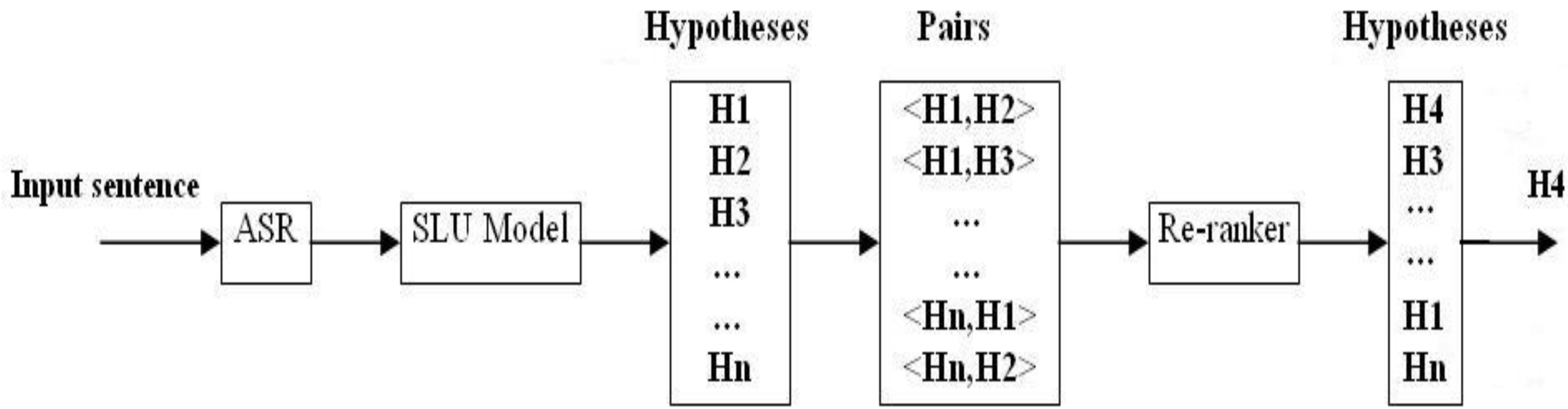
- Luna projects' Corpus Wizard of OZ

Corpus LUNA	Training set		Test set	
	words	concepts	words	concepts
Dialogs	183		67	
Turns	1,019		373	
Tokens	8,512	2,887	2,888	984
Vocabulary	1,172	34	-	-
OOV rate	-	-	3.2%	0.1%

Re-ranking Model

- The FST generates the most likely concept annotations.
- These are used to build annotation pairs, $\langle s^i, s^j \rangle$.
  - positive instances if s^i *more correct* than s^j,
- The trained binary classifier decides if s^i is more accurate than s^j.
- Each candidate annotation s^i is described by a word sequence where each word is followed by its concept annotation.

Re-ranking framework



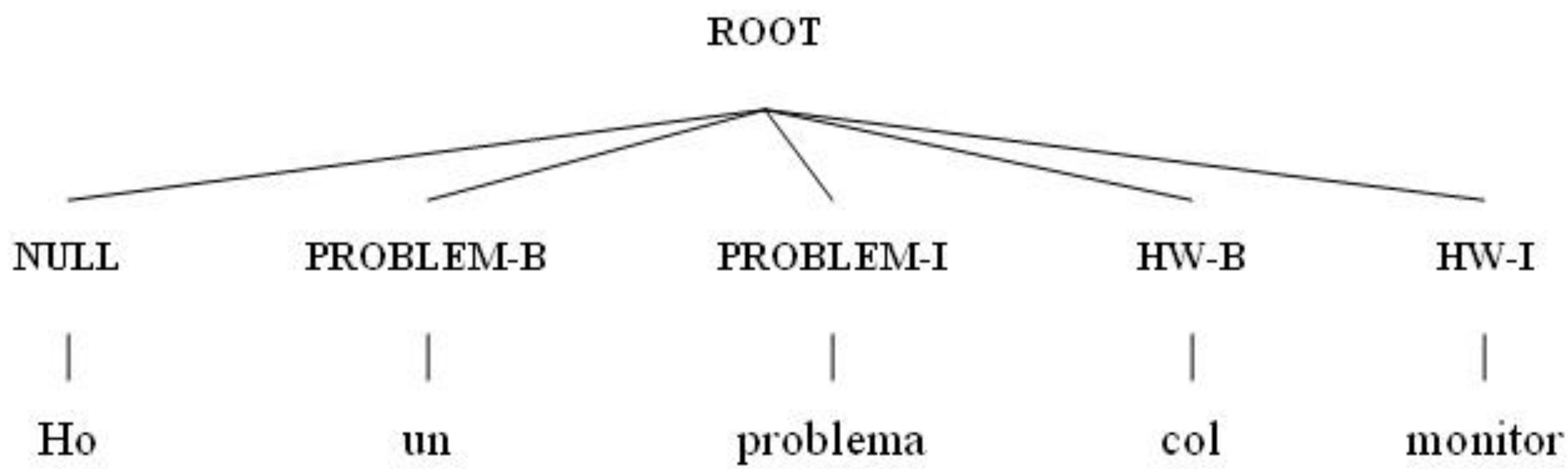
Example

- *I have a problem with the network card now*

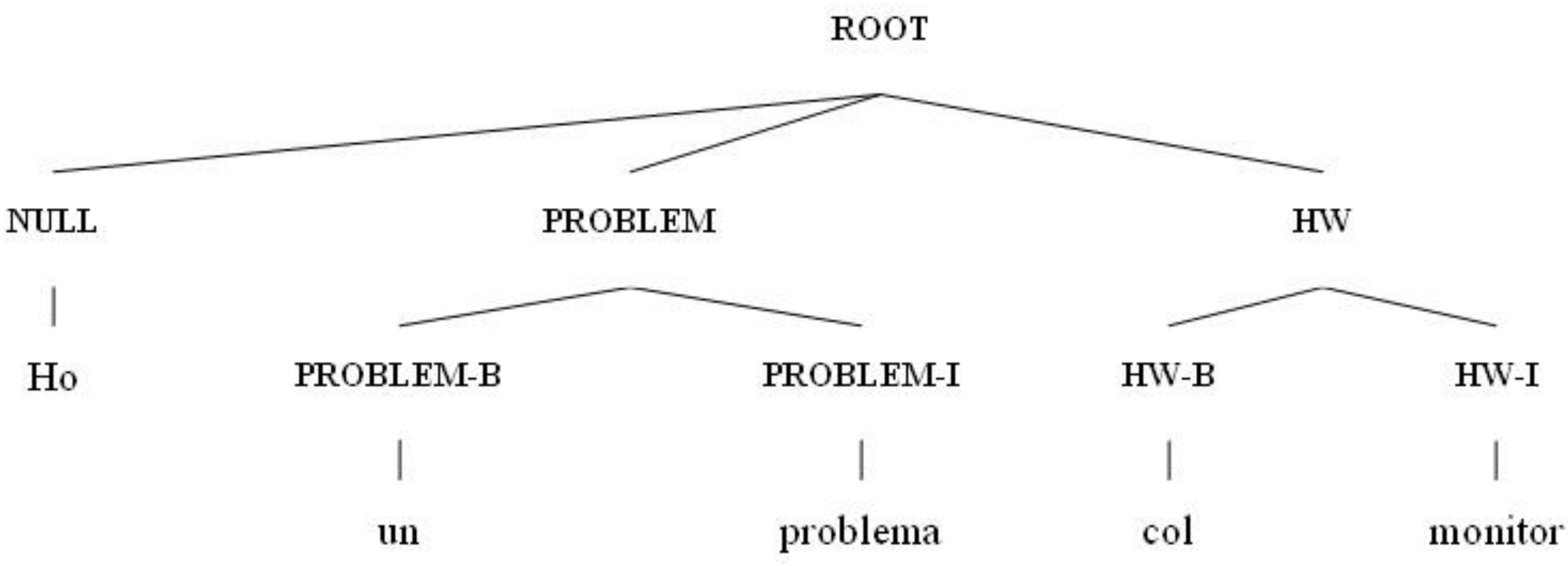
s^i: *`I`* ***`NULL`*** *`have`* ***`NULL`*** *`a`* ***`NULL`*** *`problem`* ***`PROBLEM-B`*** *`with`* ***`NULL`*** *`my`* ***`NULL`*** *`monitor`* ***`HW-B`***

s^j: *`I`* ***`NULL`*** *`have`* ***`NULL`*** *`a`* ***`NULL`*** *`problem`* ***`HW-B`*** *`with`* ***`NULL`*** *`my`* ***`NULL`*** *`monitor`*

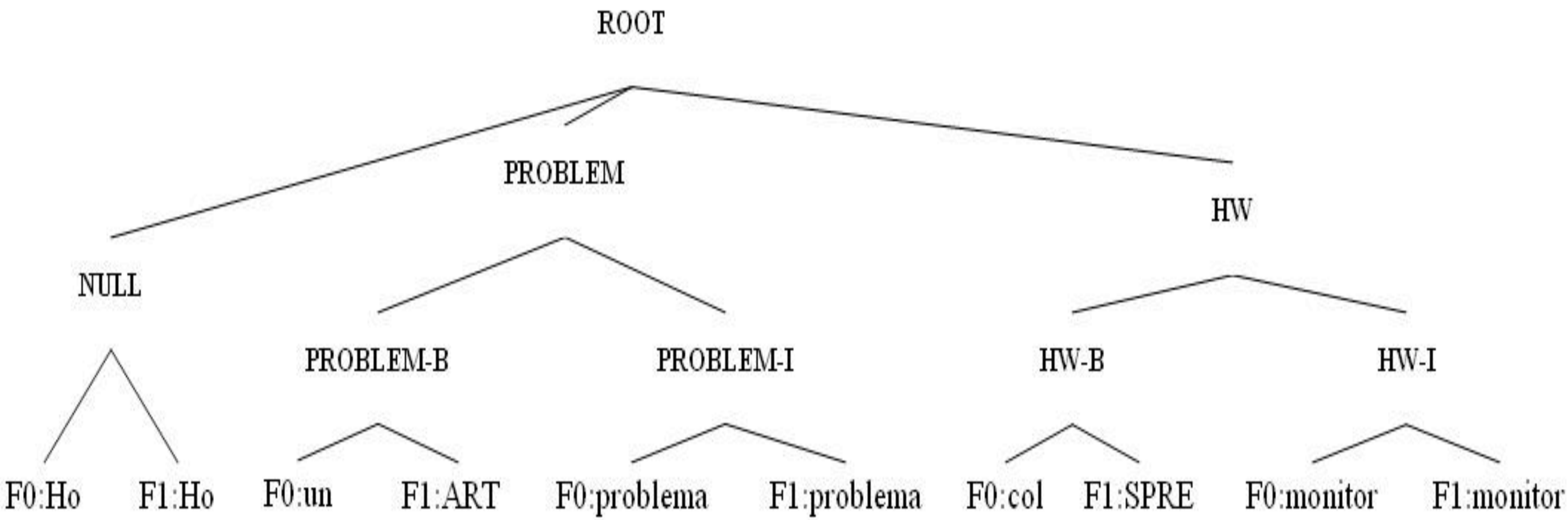
Flat tree representation



Multilevel Tree



Enriched Multilevel Tree



Results

Model	Concept Error Rate
SVMs	26.7
FSA	23.2
FSA+Re-Ranking	16.01

≈ 30% of error reduction of the best model

Structured Perceptron

Inputs: Training set (x_i, y_i) for $i = 1 \ldots n$

Initialization: $\mathbf{W} = 0$

Define: $F(x) = \mathrm{argmax}_{y \in \mathbf{GEN}(x)} \mathbf{\Phi}(x, y) \cdot \mathbf{W}$

Algorithm: For $t = 1 \ldots T, i = 1 \ldots n$

$z_i = F(x_i)$

If $(z_i \neq y_i)$ $\quad \mathbf{W} = \mathbf{W} + \mathbf{\Phi}(x_i, y_i) - \mathbf{\Phi}(x_i, z_i)$

Output: Parameters $\mathbf{W}$

Structured Output Prediction with Structural Support Vector Machines

Thorsten Joachims

Cornell University

Department of Computer Science

Joint work with
T. Hofmann, I. Tsochantaridis, Y. Altun (Brown/Google/TTI)
T. Finley, R. Elber, Chun-Nam Yu, Yisong Yue, F. Radlinski
P. Zigoris, D. Fleisher (Cornell)

Supervised Learning

- **Assume:** Data is i.i.d. from

$$P(X, Y)$$

- **Given:** Training sample

$$S = ((x_1, y_1), \ldots, (x_n, y_n))$$

- **Goal:** Find function from input space X to output space Y

$$h : X \longrightarrow Y$$

Complex objects

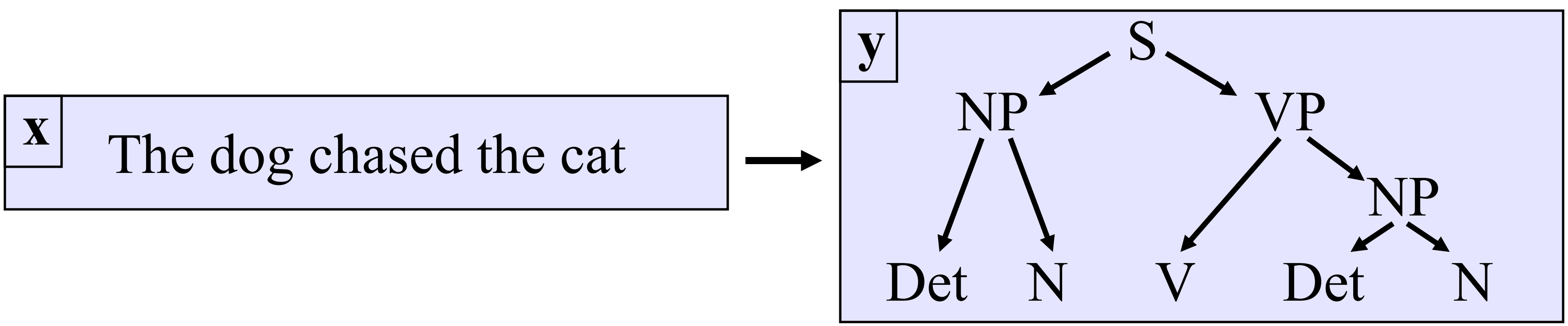
with low risk / prediction error

$$R(h) = \int \Delta(h(x), y) dP(X, Y)$$

- **Methods:** Kernel Methods, SVM, Boosting, etc.

Examples of Complex Output Spaces

- **Natural Language Parsing**
  - Given a sequence of words *x*, predict the parse tree *y*.
  - Dependencies from structural constraints, since *y* has to be a tree.



Examples of Complex Output Spaces

- **Protein Sequence Alignment**
  - Given two sequences $x=(s,t)$, predict an alignment y.
  - Structural dependencies, since prediction has to be a valid global/local alignment.

x

s=(ABJLHBNJYAUGAI)

t=(BHJKBNYGU)

$\longrightarrow$

y

```
AB-JLHBNJYAUGAI
 |  ||  ||  |||
 BHJK-BN-YGU
```

Examples of Complex Output Spaces

- **Information Retrieval**
  - Given a query x, predict a ranking *y*.
  - Dependencies between results (e.g. avoid redundant hits)
  - Loss function over rankings (e.g. AvgPrec)

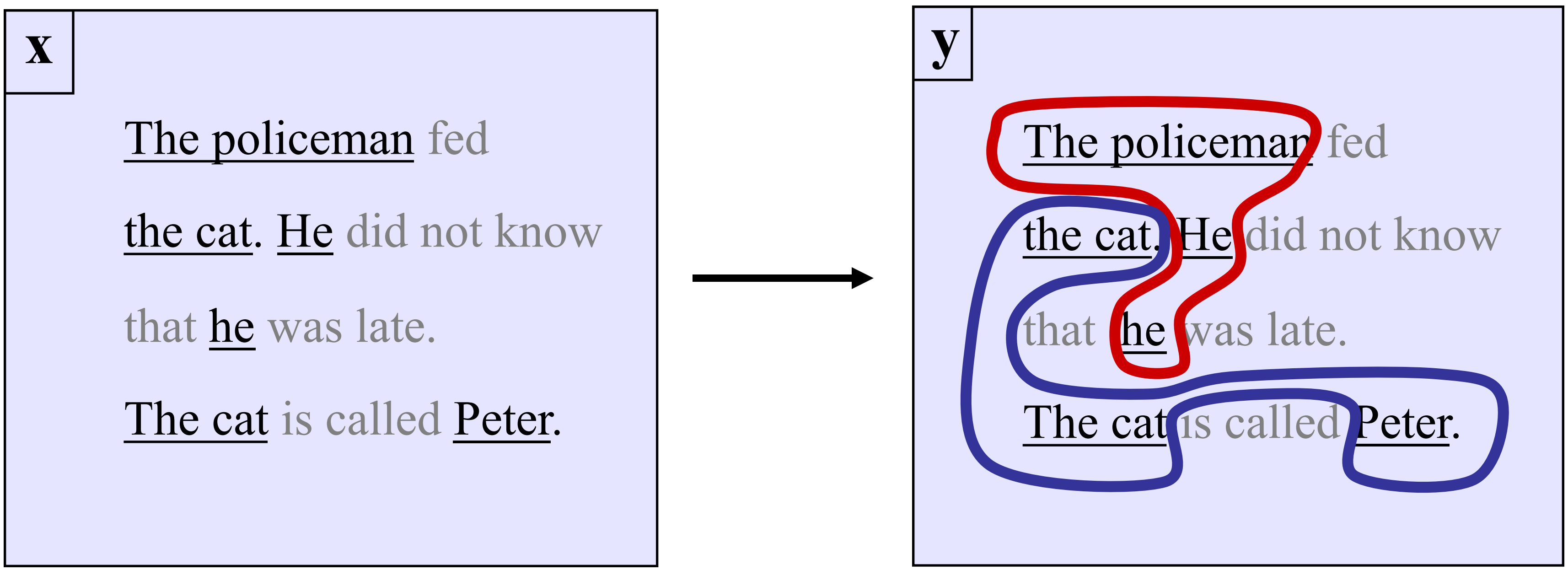
x SVM → **y**

1. Kernel-Machines
2. SVM-Light
3. Learning with Kernels
4. SV Meppen Fan Club
5. Service Master & Co.
6. School of Volunteer Management
7. SV Mattersburg Online

…

Examples of Complex Output Spaces

- **Noun-Phrase Co-reference**
  - Given a set of noun phrases *x*, predict a clustering *y*.
  - Structural dependencies, since prediction has to be an equivalence relation.
  - Correlation dependencies from interactions.



Examples of Complex Output Spaces

- **and many many more:**
  - Sequence labeling (e.g. part-of-speech tagging, named-entity recognition) [Lafferty et al. 01, Altun et al. 03]
  - Collective classification (e.g. hyperlinked documents) [Taskar et al. 03]
  - Multi-label classification (e.g. text classification) [Finley & Joachims 08]
  - Binary classification with non-linear performance measures (e.g. optimizing F1-score, avg. precision) [Joachims 05]
  - Inverse reinforcement learning / planning (i.e. learn reward function to predict action sequences) [Abbeel & Ng 04]

Overview

- **Task: Discriminative learning with complex outputs**
- **Related Work**
  - SVM algorithm for complex outputs
    - Predict trees, sequences, equivalence relations, alignments
    - General non-linear loss functions
    - Generic formulation as convex quadratic program
  - Training algorithms
    - n-slack vs. 1-slack formulation
    - Correctness and sparsity bound
  - Applications
  - Sequence alignment for protein structure prediction [w/ Chun-Nam Yu]
  - Diversification of retrieval results in search engines [w/ Yisong Yue]
  - Supervised clustering [w/ Thomas Finley]
- **Conclusions**

Why Discriminative Learning for Structured Outputs?

Precision/Recall Break-Even Point	Naïve Bayes	Linear SVM
Reuters	**72.1**	**87.5**
WebKB	**82.0**	**90.3**
Ohsumed	**62.4**	**71.6**

- **Im… it!**
  - … er 06]
  - …
- **Di…**
  - … ification
- **Improve upon prediction accuracy of existing generative methods!**
  - Natural language parsing: generative models like probabilistic context-free grammars
  - SVM outperforms naïve Bayes for text classification [Joachims, 1998] [Dumais et al., 1998]
- **More flexible models!**
  - Avoid generative (independence) assumptions
  - Kernels for structured input spaces and non-linear functions

Related Work

- **Generative training (i.e. model P(Y,X))**
  - Hidden-Markov models
  - Probabilistic context-free grammars
  - Markov random fields
  - etc.
- **Discriminative training (i.e. model P(Y|X) or minimize risk)**
  - Multivariate output regression [Izeman, 1975] [Breiman & Friedman, 1997]
  - Kernel Dependency Estimation [Weston et al. 2003]
  - Transformer networks [LeCun et al, 1998]
  - Conditional HMM [Krogh, 1994]
  - Conditional random fields [Lafferty et al., 2001]
  - Perceptron training of HMM [Collins, 2002]
  - Maximum-margin Markov networks [Taskar et al., 2003]
  - Structural SVMs [Altun et al. 03] [Joachims 03] [TsoHoJoAl04]

Overview

- **Task: Discriminative learning with complex outputs**
- **Related Work**
- **SVM algorithm for complex outputs**
  - Predict trees, sequences, equivalence relations, alignments
  - General non-linear loss functions
  - Generic formulation as convex quadratic program
- **Training algorithms**
  - n-slack vs. 1-slack formulation
  - Correctness and sparsity bound
- **Applications**
  - Sequence alignment for protein structure prediction [w/ Chun-Nam Yu]
  - Diversification of retrieval results in search engines [w/ Yisong Yue]
  - Supervised clustering [w/ Thomas Finley]
- **Conclusions**

Classification

- **Training Examples:** $(\mathbf{x}$
- **Hypothesis Space:** $h(\mathbf{x}$
- **Training:** Find hyperpl

Dual Opt. Problem:

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j (\mathbf{x}_i \cdot \mathbf{x}_j)$$

$$\text{s.t.:} \quad \cancel{\sum_{i=1}^{n} y_i \alpha_i = 0}$$

$$\forall_{i=1}^{n} : 0 \leq \alpha_i \leq \frac{C}{n}$$

$$\mathbf{w}^* = \sum_{i=1}^{n} \alpha_i^* y_i \mathbf{x}_i$$

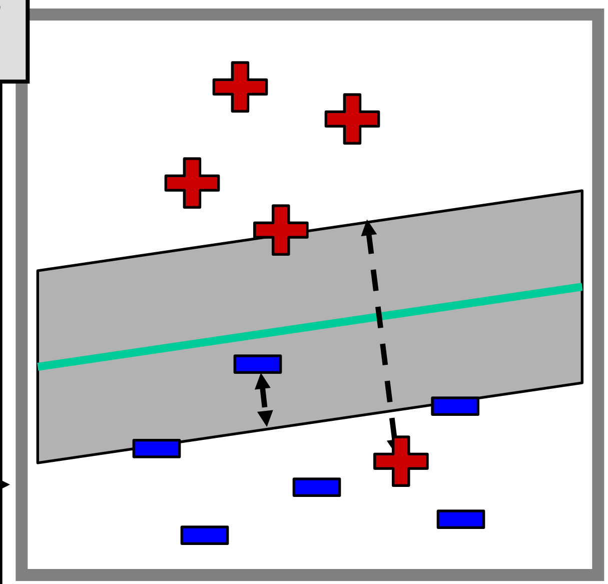
Primal Opt. Problem:

$$\min_{\mathbf{w},\xi \geq 0, b} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} + \frac{C}{n} \sum_{i=1}^{n} \xi_i$$

$$s.t. \quad y_1(\mathbf{w}^T \mathbf{x}_1 + \cancel{b}) \geq 1 - \xi_1$$

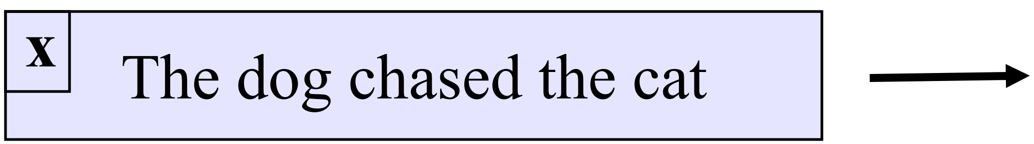
$$\ldots$$

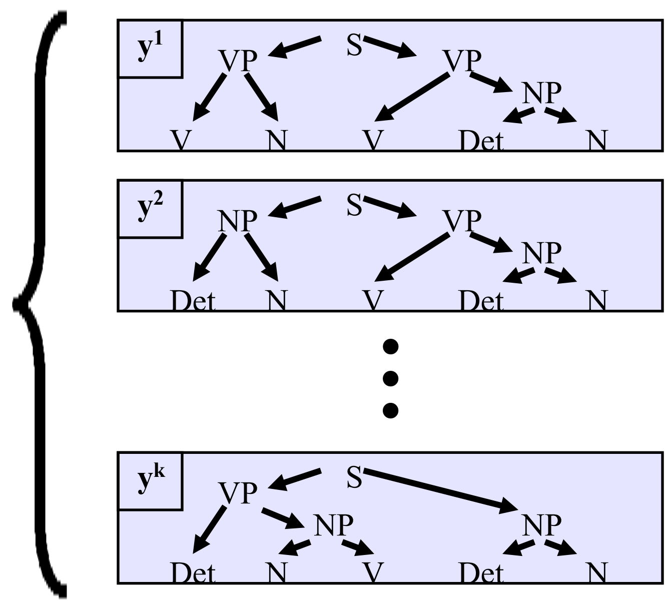
$$y_n(\mathbf{w}^T \mathbf{x}_n + \cancel{b}) \geq 1 - \xi_n$$



Challenges in Discriminative Learning with Complex Outputs

- **Approach: view as multi-class classification task**
  - Every complex output $y^i \in Y$ is one class
- **Problems:**
  - Exponentially many classes!
    - How to predict efficiently?
    - How to learn efficiently?
  - Potentially huge model!
    - Manageable number of features?





Training: Find $<\vec{w}_1, \ldots, \vec{w}_k>$ that solve

$$\min_{\vec{w}_1,\ldots,\vec{w}_n,\vec{\xi}} \quad \sum_{i=1}^{k} \vec{w}_i^2 + \frac{C}{n} \sum_{i=1}^{n} \xi_i$$

$$s.t. \quad \forall j \neq y_1 : \vec{w}_{y_1}^T \vec{x}_1 \geq \vec{w}_j^T \vec{x}_1 + 1 - \xi_1$$

$$\ldots$$

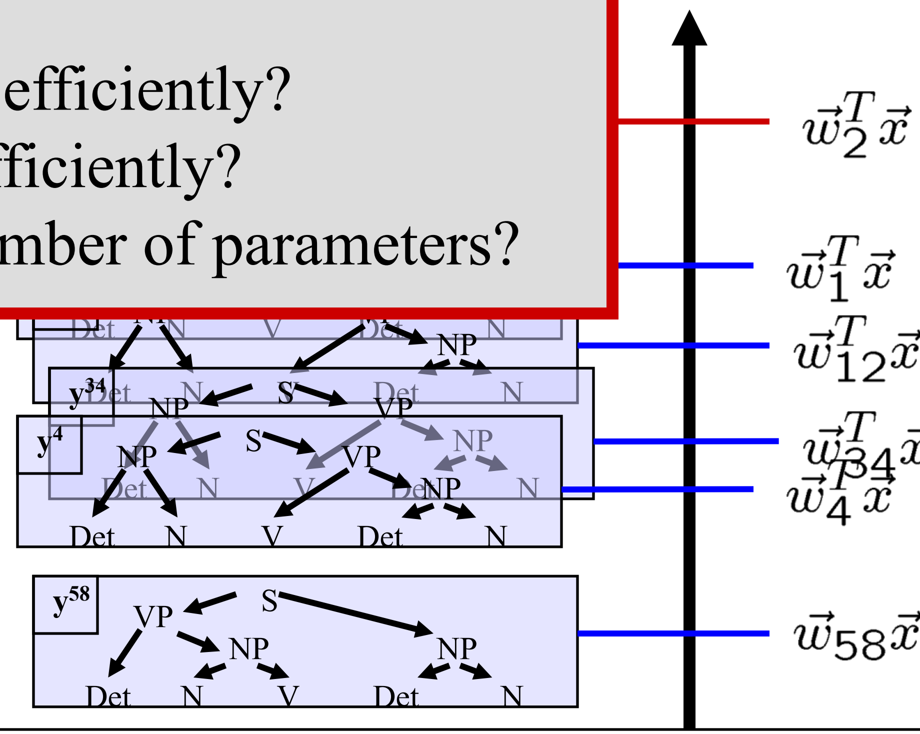
$$\forall j \neq y_n : \vec{w}_{y_n}^T \vec{x}_n \geq \vec{w}_j^T \vec{x}_n + 1 - \xi_n$$

- $\ldots, k\}$
-

Problems

- How to predict efficiently?
- How to learn efficiently?
- Manageable number of parameters?

x The dog chased the cat



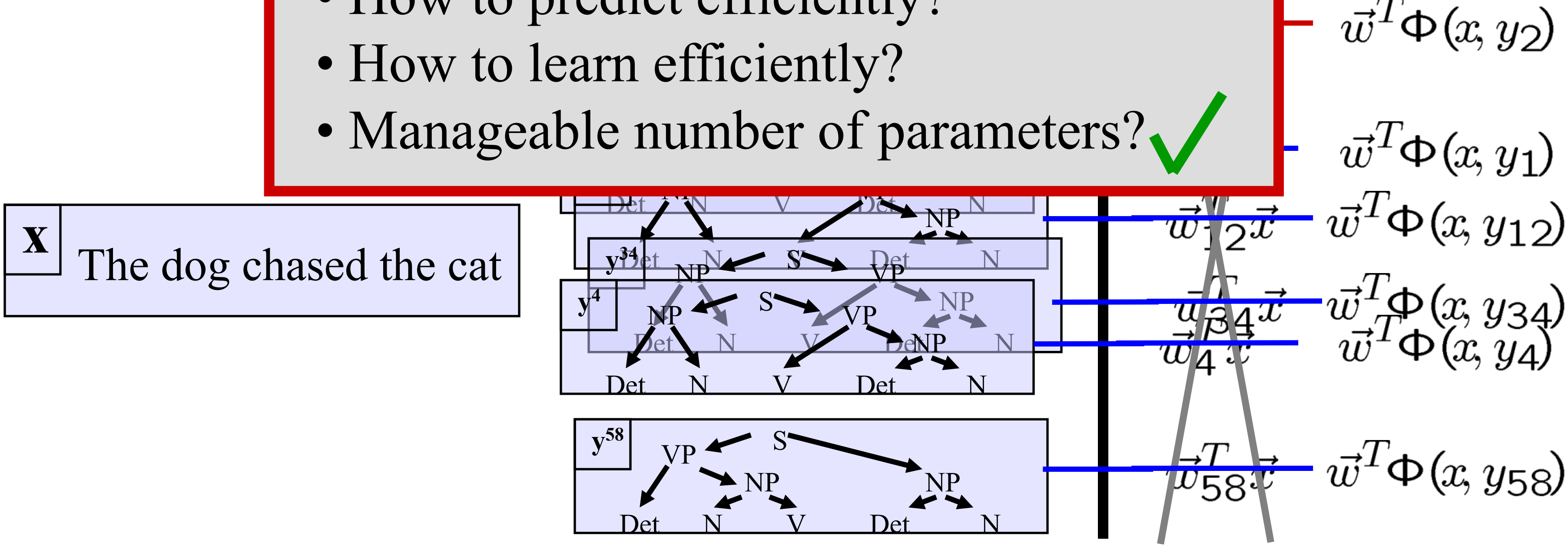
Joint Feature Map

- **Feature vector $\Phi(x, y)$ that describes match between *x* and *y***
- **Learn single weight vector and rank by $\vec{w}^T\Phi(x, y)$**

$$h(\vec{x}) = argmax_{y \in Y}\left[\vec{w}^T\Phi(x, y)\right]$$

Problems

- How to predict efficiently?
- How to learn efficiently?
- Manageable number of parameters? ✓



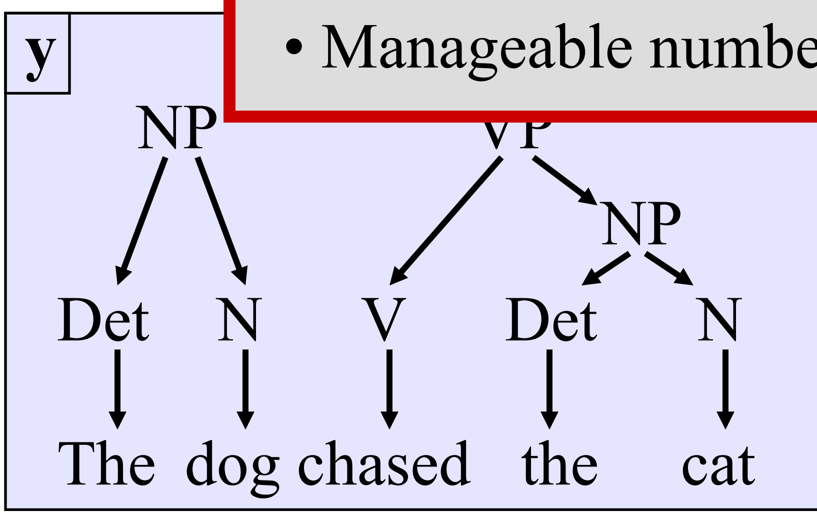
Joint Feature Map for Trees

- **Weighted Context Free Grammar**
  - Each rule (e.g. $S \rightarrow NP\ VP$) has a weight
  - Score of a tree is the sum of its weights
  - Find highest scoring tree $h(\vec{x}) = argmax_{y \in Y}\left[\vec{w}^T \Phi(x,y)\right]$

CKY Parser

x The

$f : X$



Problems
- How to predict efficiently? ✓
- How to learn efficiently?
- Manageable number of parameters? ✓

$\rightarrow NP\ VP$
$\rightarrow NP$
$\rightarrow Det\ N$
$\rightarrow V\ NP$

$\mathbf{\Phi(x,y)} = $

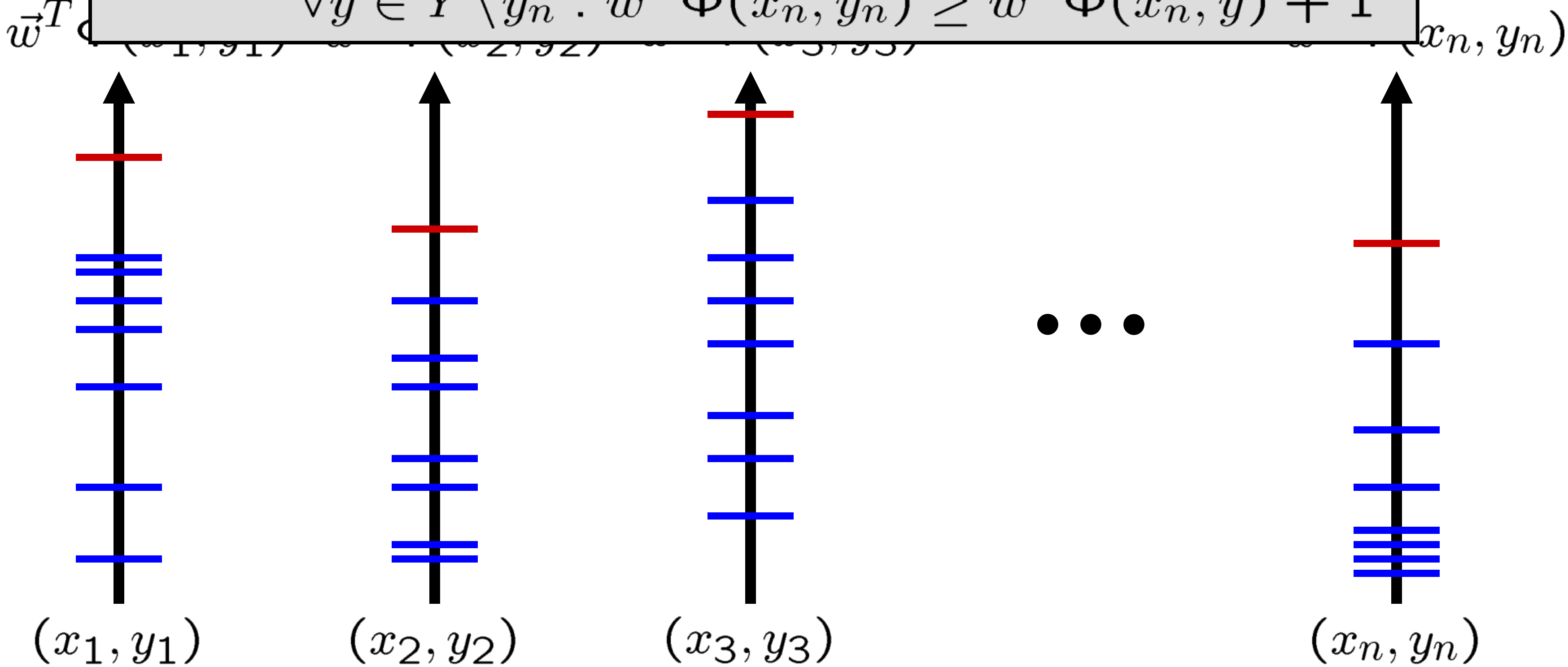
0	$Det \rightarrow dog$
2	$Det \rightarrow the$
1	$N \rightarrow dog$
1	$V \rightarrow chased$
1	$N \rightarrow cat$

Structural Support Vector Machine

Hard-margin optimization problem:

$$\min_{\vec{w}} \quad \frac{1}{2}\vec{w}^T\vec{w}$$

$$
\begin{aligned}
s.t. \quad & \forall y \in Y \setminus y_1 : \vec{w}^T\Phi(x_1, y_1) \geq \vec{w}^T\Phi(x_1, y) + 1 \\
& \dots \\
& \forall y \in Y \setminus y_n : \vec{w}^T\Phi(x_n, y_n) \geq \vec{w}^T\Phi(x_n, y) + 1
\end{aligned}
$$



Loss Functions: Soft-Margin Struct SVM

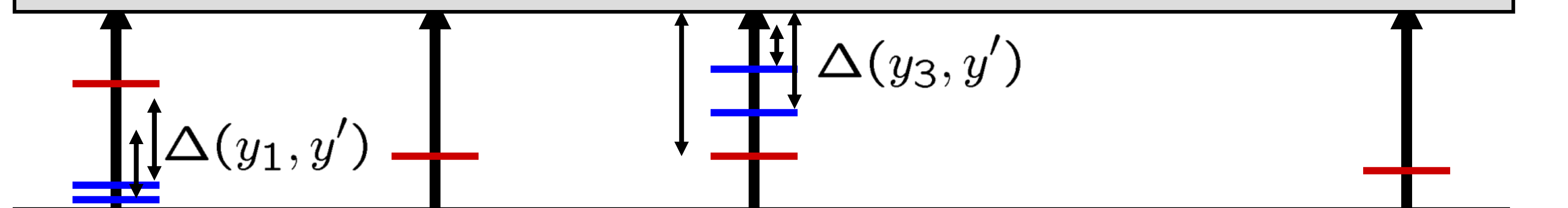
Soft-margin optimization problem:

$$\min_{\vec{w},\vec{\xi}} \quad \frac{1}{2}\vec{w}^T\vec{w} + \frac{C}{n}\sum_{i=1}^{n}\xi_i$$

$$s.t. \quad \forall y \in Y \backslash y_1 : \vec{w}^T\Phi(x_1,y_1) \geq \vec{w}^T\Phi(x_1,y) + \Delta(y_1,y) - \xi_1$$

$$\ldots$$

$$\forall y \in Y \backslash y_n : \vec{w}^T\Phi(x_n,y_n) \geq \vec{w}^T\Phi(x_n,y) + \Delta(y_n,y) - \xi_n$$



Lemma: The training loss is upper bounded by

$$Err_S(h) = \frac{1}{n}\sum_{i=1}^{n}\Delta(y_i,h(\vec{x}_i)) \leq \frac{1}{n}\sum_{i=1}^{n}\xi_i$$

(x_1,y_1) (x_2,y_2) (x_3,y_3) (x_n,y_n)

Experiment: Natural Language Parsing

- **Implemention**
  - Incorporated modified version of Mark Johnson's CKY parser
  - Learned weighted CFG with $\epsilon = 0.01, C = 1$
- **Data**
  - Penn Treebank sentences of length at most 10 (start with POS)
  - Train on Sections 2-22: 4098 sentences
  - Test on Section 23: 163 sentences

Method	Test Accuracy Acc	F_1
PCFG with MLE	55.2	86.0
SVM with (1-F_1)-Loss	**58.9**	**88.5**

[TsoJoHoAl04]

  - more complex features [TaKlCoKoMa04]

Generic Structural SVM

- **Application Specific Design of Model**
  - Loss function $\Delta(y_i, y)$
  - Representation $\Phi(x, y)$

    ➔ Markov Random Fields [Lafferty et al. 01, Taskar et al. 04]
- **Prediction:**

$$\hat{y} = argmax_{y \in Y} \{ \vec{w}^T \Phi(x, y) \}$$

- **Training:**

$$\begin{aligned} \min_{\vec{w}, \vec{\xi} \geq 0} \quad & \frac{1}{2} \vec{w}^T \vec{w} + \frac{C}{n} \sum_{i=1}^{n} \xi_i \\ s.t. \quad & \forall y \in Y \backslash y_1 : \vec{w}^T \Phi(x_1, y_1) \geq \vec{w}^T \Phi(x_1, y) + \Delta(y_1, y) - \xi_1 \\ & \ldots \\ & \forall y \in Y \backslash y_n : \vec{w}^T \Phi(x_n, y_n) \geq \vec{w}^T \Phi(x_n, y) + \Delta(y_n, y) - \xi_n \end{aligned}$$

- **Applications:** Parsing, Sequence Alignment, Clustering, etc.

Reformulation of the Structural SVM QP

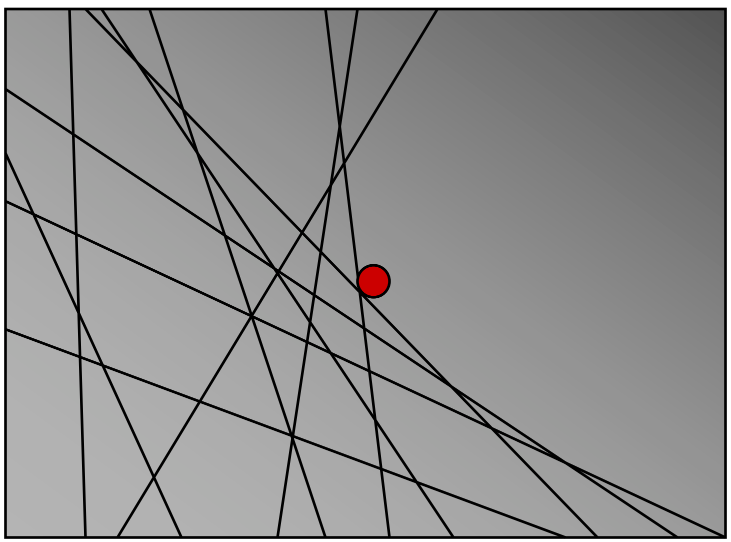
n-Slack Formulation: [TsoJoHoAl04]

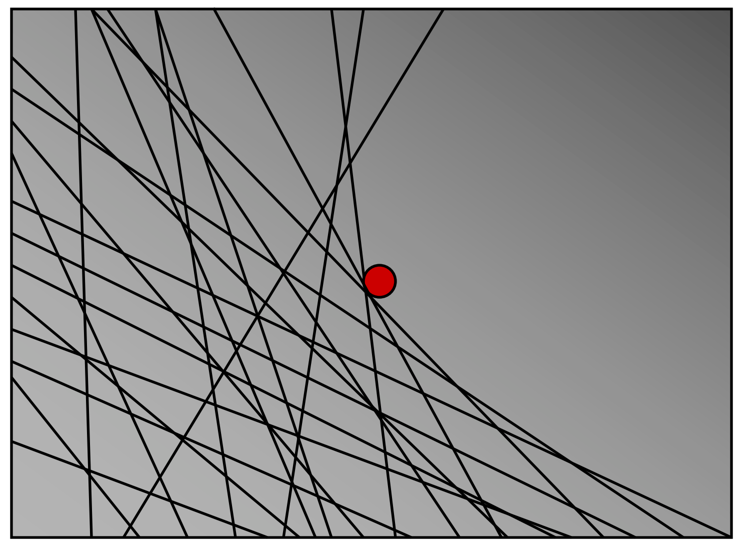
$$\min_{\vec{w},\vec{\xi}} \quad \frac{1}{2}\vec{w}^T\vec{w} + \frac{C}{n}\sum_{i=1}^{n}\xi_i$$

$$s.t. \quad \forall y' \in Y : \vec{w}^T\Phi(x_1, y_1) - \vec{w}^T\Phi(x_1, y') \geq \Delta(y_1, y) - \xi_1$$

$$\ldots$$

$$\forall y' \in Y : \vec{w}^T\Phi(x_n, y_n) - \vec{w}^T\Phi(x_n, y') \geq \Delta(y_n, y) - \xi_n$$





Reformulation of the Structural SVM QP

n-Slack Formulation: [TsoJoHoAl04]

$$\min_{\vec{w},\vec{\xi}} \quad \frac{1}{2}\vec{w}^T\vec{w} + \frac{C}{n}\sum_{i=1}^{n}\xi_i$$

$$s.t. \quad \forall y' \in Y : \vec{w}^T\Phi(x_1, y_1) - \vec{w}^T\Phi(x_1, y') \geq \Delta(y_1, y) - \xi_1$$

$$\dots$$

$$\forall y' \in Y : \vec{w}^T\Phi(x_n, y_n) - \vec{w}^T\Phi(x_n, y') \geq \Delta(y_n, y) - \xi_n$$

$\Leftrightarrow$

1-Slack Formulation: [JoFinYu08]

$$\min_{\vec{w},\xi} \quad \frac{1}{2}\vec{w}^T\vec{w} + C\xi$$

$$s.t. \quad \forall y'_1 \dots y'_n \in Y : \frac{1}{n}\sum_{i=1}^{n}\left[\vec{w}^T\Phi(x_i, y_i) - \vec{w}^T\Phi(x_i, y'_i)\right] \geq \frac{1}{n}\sum_{i=1}^{n}\left[\Delta(y_i, y'_i)\right] - \xi$$

Cutting-Plane Algorithm for Structural SVM (1-Slack Formulation)

- **Input:** $(x_1, y_1), \ldots, (x_n, y_n), C, \epsilon$
- $S \leftarrow \emptyset, \vec{w} \leftarrow 0, \xi \leftarrow 0$
- **REPEAT**
  - FOR $i = 1, \ldots, n$
    - Compute $y_i' = argmax_{y \in Y} \{\Delta(y_i, y) + \vec{w}^T \Phi(x_i, y)\}$
  - ENDFOR
  - IF $\sum_{i=1}^{n} \left[\Delta(y_i, y_i') - \vec{w}^T [\Phi(x_i, y_i) - \Phi(x_i, y_i')]\right] > \xi + \epsilon$
    - $S \leftarrow S \cup \{\vec{w}^T \frac{1}{n} \sum_{i=1}^{n} [\Phi(x_i, y_i) - \Phi(x_i, y_i')] \geq \frac{1}{n} \sum_{i=1}^{n} \Delta(y_i, y_i') - \xi\}$
    - $[\vec{w}, \xi]$ optimize StructSVM over
  - ENDIF
- **UNTIL has not changed during iteration**

Find most violated constraint

Violated by more than ε ?

Add constraint to working set

[Jo06] [JoFinYu08]

Polynomial Sparsity Bound

- **Theorem:** The cutting-plane algorithm finds a solution to the Structural SVM soft-margin optimization problem in the 1-slack formulation after adding at most

$$\left\lceil \log_2\left(\frac{\Delta}{4R^2C}\right)\right\rceil + \left\lceil\frac{16R^2C}{\varepsilon}\right\rceil$$

constraints to the working set S, so that the primal constraints are feasible up to a precision and the objective on S is optimal. The loss has to be bounded $0 \leq \Delta(y_i, y) \leq \Delta$, and $2||\Phi(x,y)|| \leq R$.

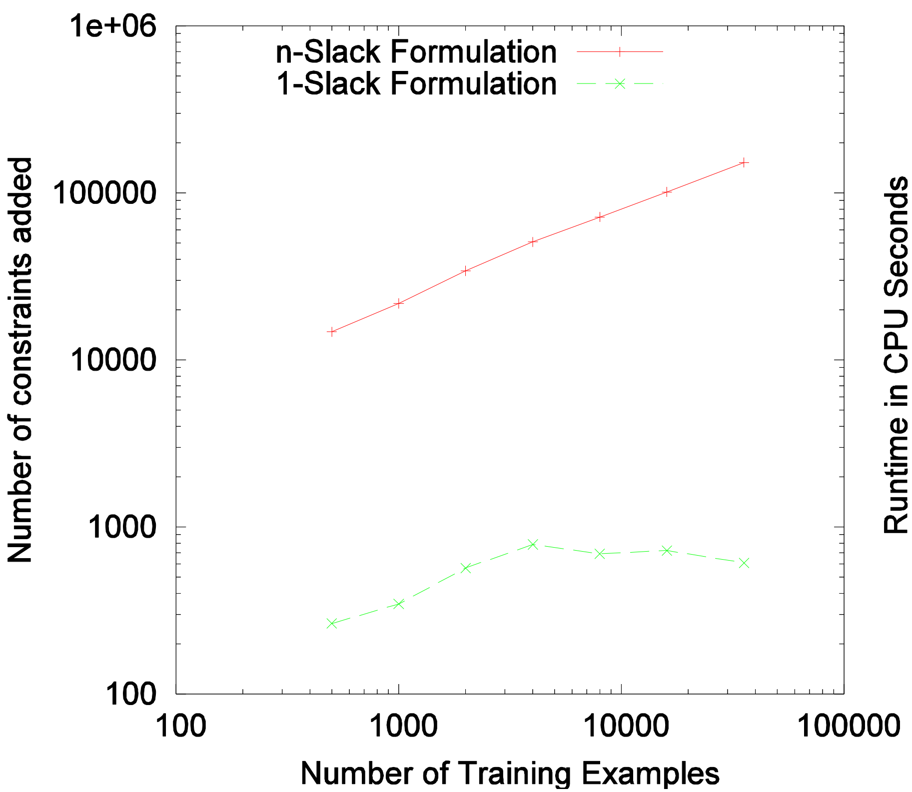
[Jo03] [Jo06] [TeoLeSmVi07] [JoFinYu08]

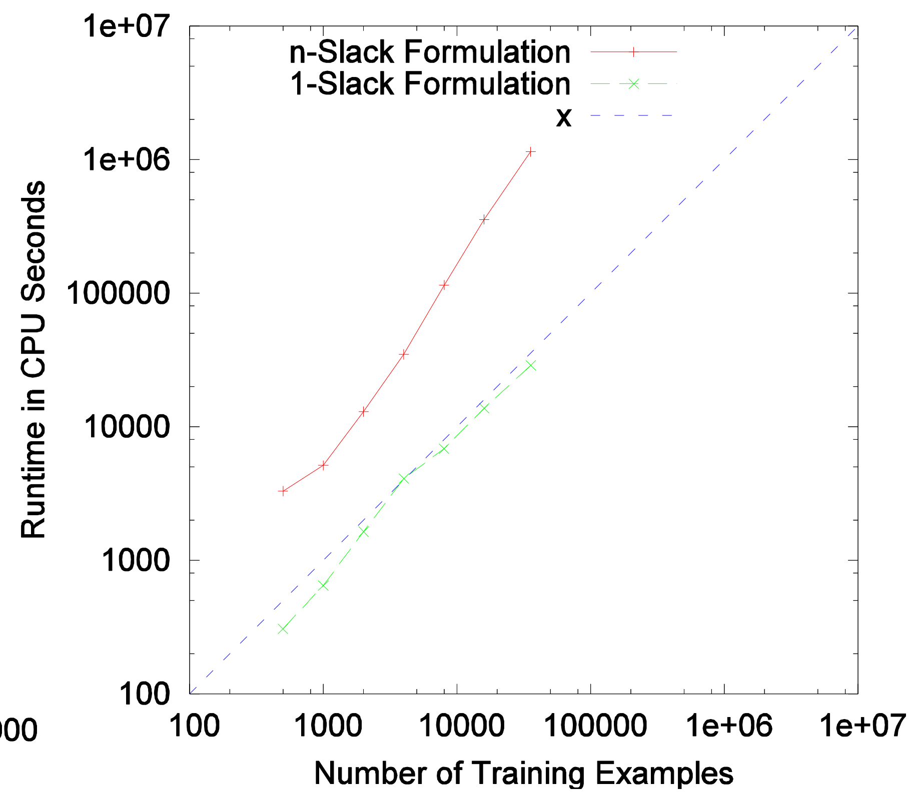
Empirical Comparison: Different Formulations

Experiment Setup:

– Part-of-speech tagging on Penn Treebank corpus

– ~36,000 examples, ~250,000 features in linear HMM model





[JoFinYu08]

Applying StructSVM to New Problem

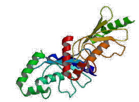
- **General**
  - SVM-struct algorithm and implementation
    http://svmlight.joachims.org
  - Theory (e.g. training-time linear in n)
- **Application specific**
  - Loss function $\Delta(y_i, y)$
  - Representation $\Phi(x, y)$
  - Algorithms to compute
    $$\hat{y} = argmax_{y \in Y} \{\vec{w}^T \Phi(x_i, y)\}$$
    $$\hat{y} = argmax_{y \in Y} \{\Delta(y_i, y) + \vec{w}^T \Phi(x_i, y)\}$$
- **Properties**
  - General framework for discriminative learning
  - Direct modeling, not reduction to classification/regression
  - “Plug-and-play”

Overview

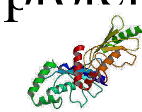
- **Task: Discriminative learning with complex outputs**
- **Related Work**
- **SVM algorithm for complex outputs**
  - Predict trees, sequences, equivalence relations, alignments
  - General non-linear loss functions
  - Generic formulation as convex quadratic program
- **Training algorithms**
  - n-slack vs. 1-slack formulation
  - Correctness and sparsity bound
- **Applications**
  - Sequence alignment for protein structure prediction [w/ Chun-Nam Yu]
  - Diversification of retrieval results in search engines [w/ Yisong Yue]
  - Supervised clustering [w/ Thomas Finley]
- **Conclusions**

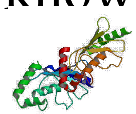
Comparative Modeling of Protein Structure

- **Goal: Predict structure from sequence**

  h("APPGEAYLQV") → 

- **Hypothesis:**
  - Amino Acid sequences for into structure with lowest energy
  - Problem: Huge search space (> 2^{100} states)
- **Approach: Comparative Modeling**
  - Similar protein sequences fold into similar shapes
    → use known shapes as templates
  - Task 1: Find a similar known protein for a new protein

    h("APPGEAYLQV", ) → yes/no
  - Task 2: Map new protein into known structure

    h("APPGEAYLQV", ) → [A→3,P→4,P→7,…]
  - Task 3: Refine structure

[Jo03, JoElGa05,YuJoEl06]

Linear Score Sequence Alignment

Method: Find alignment *y* that maximizes linear score

$$y = argmax_{y \in Y} \{score(x=(s,t), y)\}$$

Example:

– Sequences:

s=(A B C D)

t=(B A C C)

	A	B	C	D	–
A	10	0	-5	-10	-5
B	0	10	5	-10	-5
C	-5	5	10	-10	-5
D	-10	-10	-10	10	-5
–	-5	-5	-5	-5	-5

– Alignment y_1:

```
A B C D
B A C C
```

→ $score(x=(s,t),y_1) = 0+0+10-10 = 0$

– Alignment y_2:

```
- A B C D
B A C C -
```

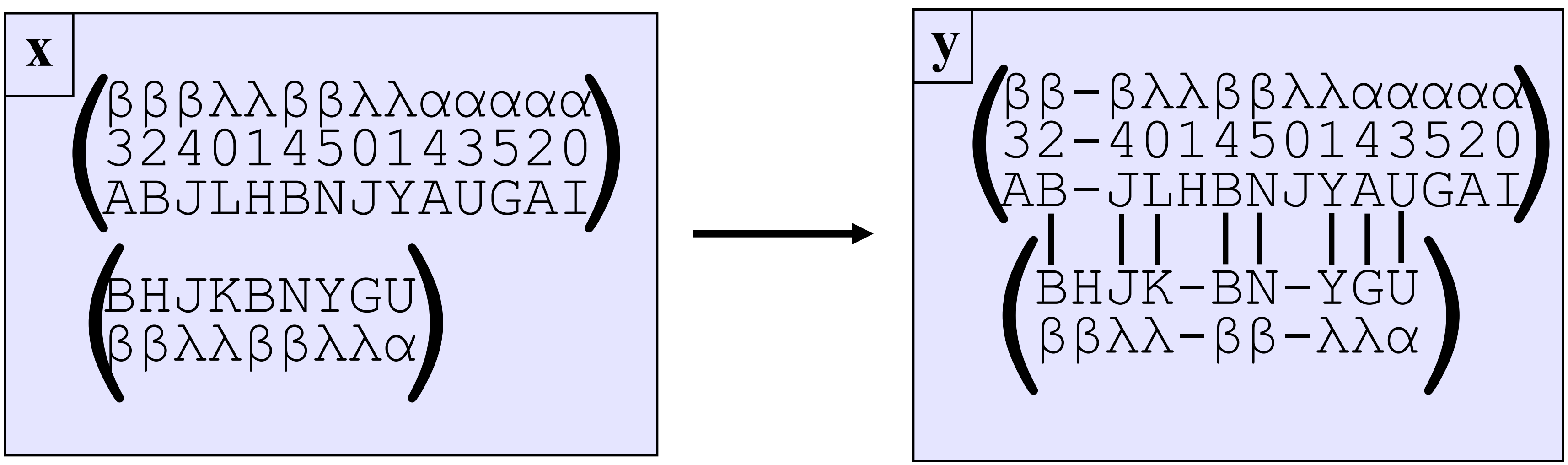
→ $score(x=(s,t),y_2) = -5+10+5+10-5 = 15$

Algorithm: Solve argmax via dynamic programming.

Predicting an Alignment

Protein Sequence to Structure Alignment (Threading)

- Given a pair *x=(s,t)* of new sequence *s* and known structure *t*, predict the alignment *y*.
- Elements of *s* and *t* are described by features, not just character identity.



[YuJoEl07]

Scoring Function for Vector Sequences

General form of linear scoring function:

$$
\begin{aligned}
score\,(\mathbf{x}{=}(\mathbf{s},\mathbf{t}),\mathbf{y}) &= \sum_i score(y_i^{\mathbf{s}}, y_i^{\mathbf{t}}) \\
&= \sum_i \mathbf{w}^T \phi(\mathbf{s},\mathbf{t},y_i) \\
&= \mathbf{w}^T \sum_i \phi(\mathbf{s},\mathbf{t},y_i) \\
&= \mathbf{w}^T \Phi(\mathbf{x},\mathbf{y})
\end{aligned}
$$

→ match/gap score can be arbitrary linear function

→ argmax can still be computed efficiently via dynamic programming

Estimation:

– Generative estimation (e.g. log-odds, hidden Markov model)

– Discriminative estimation via structural SVM

[YuJoEl07]

Loss Function and Separation Oracle

- **Loss function:** $\Delta(y_i, y)$
  - Q loss: fraction of incorrect alignments
    - Correct alignment **y**=

–	A	B	C	D
B	A	C	C	–

    - Alternate alignment **y'**=

A	–	B	C	D
B	A	C	C	–

$\rightarrow \Delta_Q(y,y')=1/3$

  - Q4 loss: fraction of incorrect alignments outside window
    - Correct alignment **y**=

–	A	B	C	D
B	A	C	C	–

    - Alternate alignment **y'**=

A	–	B	C	D
B	A	C	C	–

$\rightarrow \Delta_{Q4}(y,y')=0/3$

- **Separation oracle:** $\hat{y} = argmax_{y \in Y} \{ \Delta(y_i, y) + \vec{w}^T \Phi(x_i, y) \}$
  - Same dynamic programming algorithms as alignment

[YuJoEl07]

Experiment

- **Train set [Qiu & Elber]:**
  - 5119 structural alignments for training, 5169 structural alignments for validation of regularization parameter C
- **Test set:**
  - 29764 structural alignments from new deposits to PDB from June 2005 to June 2006.
  - All structural alignments produced by the program CE by superimposing the 3D coordinates of the proteins structures. All alignments have CE Z-score greater than 4.5.
- **Features (known for structure, SABLE predictions for sequence):**
  - Amino acid identity (A,C,D,E,F,G,H,I,K,L,M,N,P,Q,R,S,T,V,W,Y)
  - Secondary structure (α,β,λ)
  - Exposed surface area (0,1,2,3,4,5)

[YuJoEl07]

Experiment Results

Models:

- **Simple:** $\Phi(s,t,y_i) \Leftrightarrow$ (A|A; A|C; …;-|Y; α|α; α|β…; 0|0; 0|1;…)
- **Anova2:** $\Phi(s,t,y_i) \Leftrightarrow$ (Aα|Aα…; α0|α0…; A0|A0;…)
- **Tensor:** $\Phi(s,t,y_i) \Leftrightarrow$ (Aα0|Aα0; Aα0|Aα1; …)
- **Window:** $\Phi(s,t,y_i) \Leftrightarrow$ (AAA|AAA; …; ααααα|ααααα; …; 00000|00000;…)

Ability to train complex models?

Q-Score	# Features	Test
Simple	1020	39.89
Anova2	49634	44.98
Tensor	203280	42.81
Window	447016	46.30

Q-score when optimizing to Q-loss

Comparison against other methods?

Q4-score	Test
BLAST	28.44
SVM (Window)	70.71
SSALN [QiuElber]	67.30
TM-align [ZhaSko]	(85.32)

Q4-score when optimizing to Q4-loss

[YuJoEl07]

Overview

- **Task: Discriminative learning with complex outputs**
- **Related Work**
- **SVM algorithm for complex outputs**
  - Predict trees, sequences, equivalence relations, alignments
  - General non-linear loss functions
  - Generic formulation as convex quadratic program
- **Training algorithms**
  - n-slack vs. 1-slack formulation
  - Correctness and sparsity bound
- **Applications**
  - Sequence alignment for protein structure prediction [w/ Chun-Nam Yu]
  - Diversification of retrieval results in search engines [w/ Yisong Yue]
  - Supervised clustering [w/ Thomas Finley]
- **Conclusions**

Diversified Retrieval

- **Ambiguous queries:**
  - Example query: "SVM"
    - ML method
    - Service Master Company
    - Magazine
    - School of veterinary medicine
    - Sport Verein Meppen e.V.
    - SVM software
    - SVM books
  - "submodular" performance measure

    ➔ make sure each user gets at least one relevant result
- **Learning Queries:**
  - Find all information about a topic
  - Eliminate redundant information

Query: SVM

1. Kernel Machines
2. SVM book
3. SVM-light
4.
5.
6.
7.

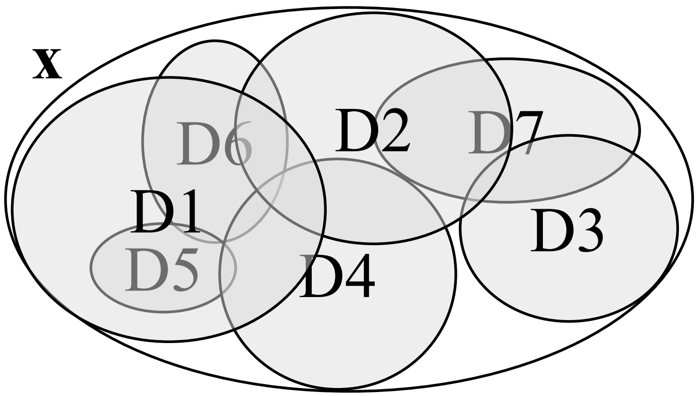
Query: SVM

1. Kernel Machines
2. Service Master Co
3. SV Meppen
4. UArizona Vet. Med.
5. SVM-light
6. Intro to SVM
7. …

[YueJo08]

Approach

- **Prediction Problem:**
  - Given set **x**, predict size *k* subset **y** that satisfies most users.
- **Approach: Topic Red. ¼ Word Red. [SwMaKi08]**



→ **y** = { D1, D2, D3, D4 }

  - Weighted Max Coverage: $\mathbf{y} = \operatorname*{argmax}_{y \subset x, |y|=k} \left\{ \sum_{w \in \cup(y)} score(w) \right\}$
  - Greedy algorithm is 1-1/e approximation [Khuller et al 97]

→ **Learn the benefit weights:** $score(w) = \mathbf{w}^T \phi(w, x)$

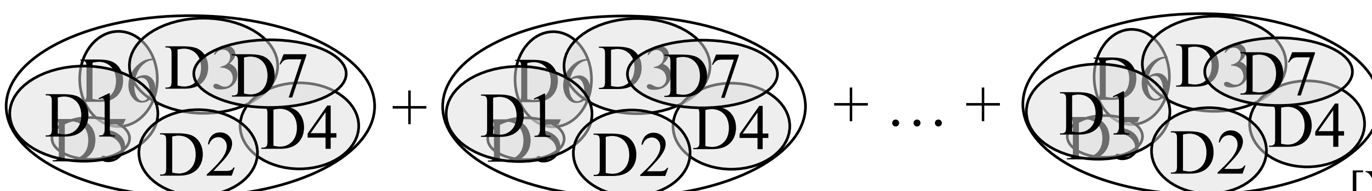
[YueJo08]

Features Describing Word Importance

- **How important is it to cover word w**
    - w occurs in at least X% of the documents in x
    - w occurs in at least X% of the titles of the documents in x
    - w is among the top 3 TFIDF words of X% of the documents in x
    - w is a verb

  → Each defines a feature in $\phi(w, x)$
- **How well a document d covers word w**
    - w occurs in d
    - w occurs at least k times in d
    - w occurs in the title of d
    - w is among the top k TFIDF words in d

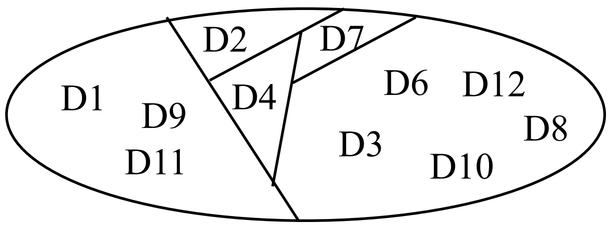
  → Each defines a separate vocabulary and scoring function



[YueJo08]

Loss Function and Separation Oracle

- **Loss function:** $\Delta(y_i, y)$
  - Popularity-weighted percentage of subtopics not covered in **y**
    → More costly to miss popular topics
  - Example:



$$\Delta(y_i, \{D1, D10\}) = 3/12$$
$$\Delta(y_i, \{D2, D7\}) = 10/12$$

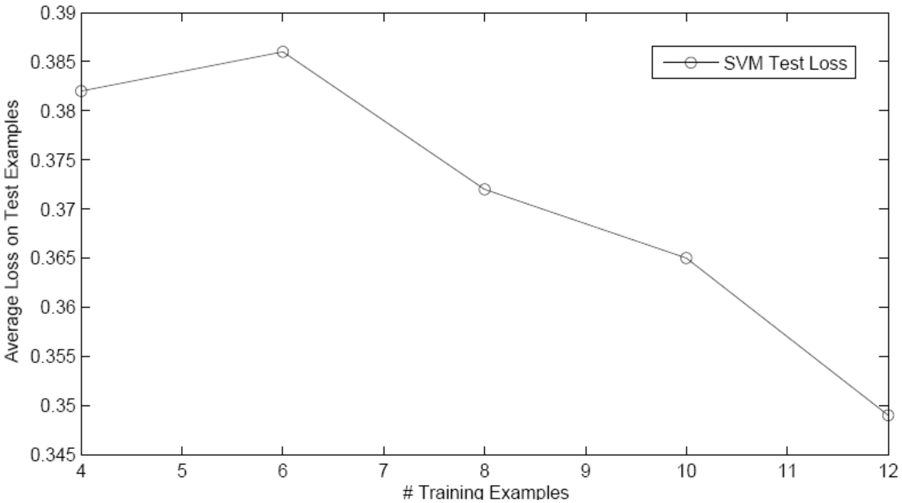
- **Separation oracle:** $\hat{y} = argmax_{y \in Y}\{\Delta(y_i, y) + \vec{w}^T \Phi(x_i, y)\}$
  - Again a weighted max coverage problem
    → add artificial word for each subtopic with percentage weight
  - Greedy algorithm is 1-1/e approximation [Khuller et al 97]

[YueJo08]

Experiments

- **Data:**
  - TREC 6-8 Interactive Track
  - Relevant documents manually labeled by subtopic
  - 17 queries (~700 documents), 12/4/1 training/validation/test
  - Subset size k=5, two feature sets (div, div2)
- **Results:**

Method	Loss
Random	0.469
Okapi	0.472
Unweighted Model	0.471
Essential Pages	0.434
$\mathrm{SVM}^{\Delta}_{div}$	0.349
$\mathrm{SVM}^{\Delta}_{div2}$	0.382

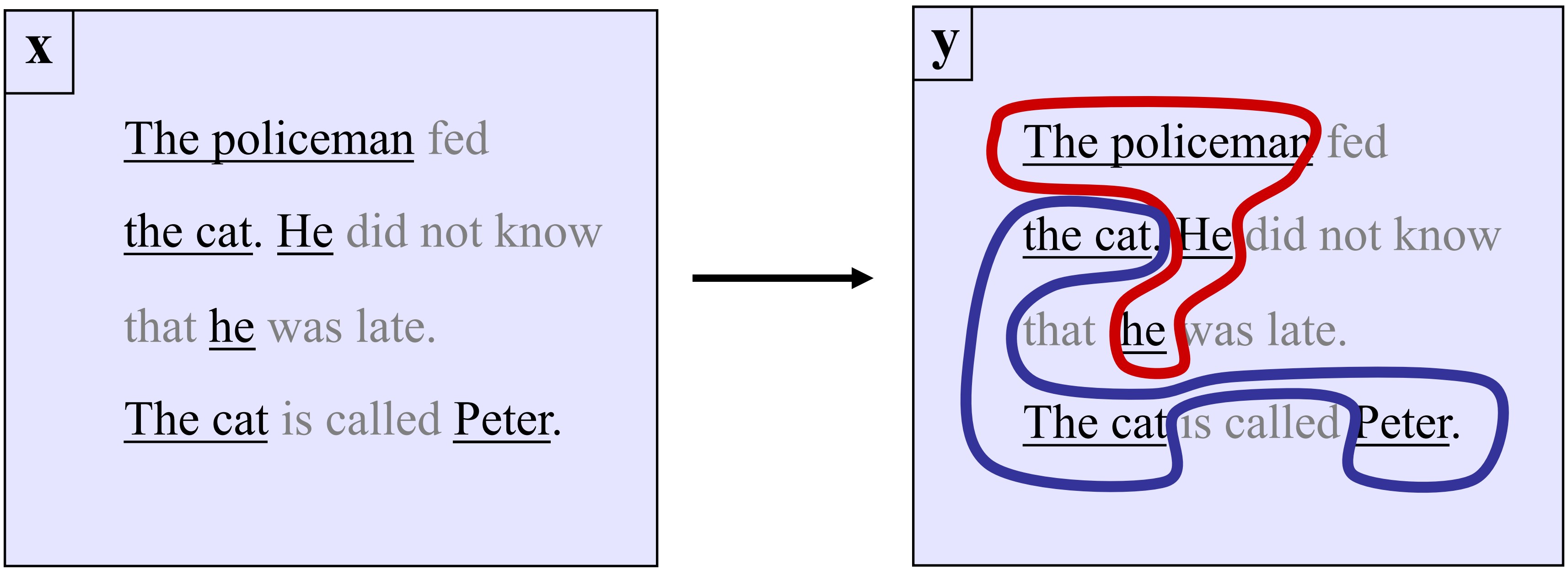


Overview

- **Task: Discriminative learning with complex outputs**
- **Related Work**
- **SVM algorithm for complex outputs**
  - Predict trees, sequences, equivalence relations, alignments
  - General non-linear loss functions
  - Generic formulation as convex quadratic program
- **Training algorithms**
  - n-slack vs. 1-slack formulation
  - Correctness and sparsity bound
- **Applications**
  - Sequence alignment for protein structure prediction [w/ Chun-Nam Yu]
  - Diversification of retrieval results in search engines [w/ Yisong Yue]
  - Supervised clustering [w/ Thomas Finley]
- **Conclusions**

Learning to Cluster

- **Noun-Phrase Co-reference**
  - Given a set of noun phrases *x*, predict a clustering *y*.
  - Structural dependencies, since prediction has to be an equivalence relation.
  - Correlation dependencies from interactions.



Struct SVM for Supervised Clustering

- **Representation**

$$\vec{w}^T\vec{x}_i \quad \begin{matrix} 1.0 & 2.5 & 0.0 & 0.0 & -2.1 & -0.1 & 0.0 \\ 0.0 & 1.0 & 3.0 & 0.1 & 0.1 & 0.3 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.3 & 2.2 & -0.9 & 0.0 \\ 0.9 & -0.4 & -0.1 & 1.0 & -1.1 & -0.1 & -2.0 \\ 0.0 & 0.0 & 0.0 & -0.3 & 1.0 & 2.3 & 0.0 \\ 0.1 & 0.1 & 0.0 & -0.1 & 0.1 & 1.0 & 1.8 \\ -1.0 & -0.7 & 0.0 & -0.2 & -1.1 & 0.1 & 1.0 \end{matrix} \longrightarrow \mathbf{y} \quad \begin{matrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{matrix}$$

  - $\{0, 1\}$
  - nd
  -
- **Los**
  - $\Delta(y, y') = ||y - y'||_1$

$$\mathbf{y} \quad \begin{matrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{matrix} \qquad \mathbf{y'} \quad \begin{matrix} 1 & 1 & 1 & \mathbf{0} & 0 & 0 & 0 \\ 1 & 1 & 1 & \mathbf{0} & 0 & 0 & 0 \\ 1 & 1 & 1 & \mathbf{0} & 0 & 0 & 0 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{matrix}$$

- **Predic**
  - $\hat{y} =$ … $\sum y_{ij}(\vec{w}^T x_{ij})\}$
  - NP … orlica, 2003]
- **Find m**
  - $\hat{y} =$
  - NP … orlica, 2003]

[FiJo05]

Summary and Conclusions

- **Learning to predict complex output**
  - Directly model machine learning application end-to-end
- **An SVM method for learning with complex outputs**
  - General method, algorithm, and theory
  - Plug in representation, loss function, and separation oracle
  - More details and further work:
    - Diversified retrieval [Yisong Yue, ICML08]
    - Sequence alignment [Chun-Nam Yu, RECOMB07, JCB08]
    - Supervised k-means clustering [Thomas Finley, forthcoming]
    - Approximate inference and separation oracle [Thomas Finley, ICML08]
    - Efficient kernelized structural SVMs [Chun-Nam Yu, KDD08]
- **Software: SVM^{struct}**
  - General API
  - Instances for sequence labeling, binary classification with non-linear loss, context-free grammars, diversified retrieval, sequence alignment, ranking
  - `http://svmlight.joachims.org/`

PART II: Basics of Natural Language Processing

Part-of-Speech tagging

- Given a sentence $W_1 \ldots W_n$ and a tagset of lexical categories, find the most likely tag $T_1 .. T_n$ for each word in the sentence
- Example

  Secretariat/NNP is/VBZ expected/VBN to/TO race/VB tomorrow/NN

  People/NNS continue/VBP to/TO inquire/VB the/DT reason/NN for/IN the/DT race/NN for/IN outer/JJ space/NN
- Note that many of the words may have unambiguous tags
  - But enough words are either ambiguous or unknown that it's a nontrivial task

Part Of Speech (POS) Tagging

- Annotate each word in a sentence with a part-of-speech.

  I ate the spaghetti with meatballs.
  Pro V Det N Prep N

  John saw the saw and decided to take it to the table.
  PN V Det N Con V Part V Pro Prep Det N

- Useful for subsequent syntactic parsing and word sense disambiguation.

PTB Tagset (36 main tags + punctuation tags)

CC	Coordinating conjunction
CD	Cardinal number
DT	Determiner
EX	Existential *there*
FW	Foreign word
IN	Preposition or subordinating conjunction
JJ	Adjective
JJR	Adjective, comparative
JJS	Adjective, superlative
LS	List item marker
MD	Modal
NN	Noun, singular or mass
NNS	Noun, plural
NP	Proper noun, singular
NPS	Proper noun, plural
PDT	Predeterminer
POS	Possessive ending
PP	Personal pronoun
PP$	Possessive pronoun
RB	Adverb
RBR	Adverb, comparative
RBS	Adverb, superlative
RP	Particle
SYM	Symbol
TO	*to*
UH	Interjection
VB	Verb, base form
VBD	Verb, past tense
VBG	Verb, gerund or present participle
VBN	Verb, past participle
VBP	Verb, non-3rd person singular present
VBZ	Verb, 3rd person singular present
WDT	Wh-determiner
WP	Wh-pronoun
WP$	Possessive wh-pronoun
WRB	Wh-adverb

Solution

- Text Classifier:
  - Tags categories
  - Features windows of words around the target word
  - N-grams

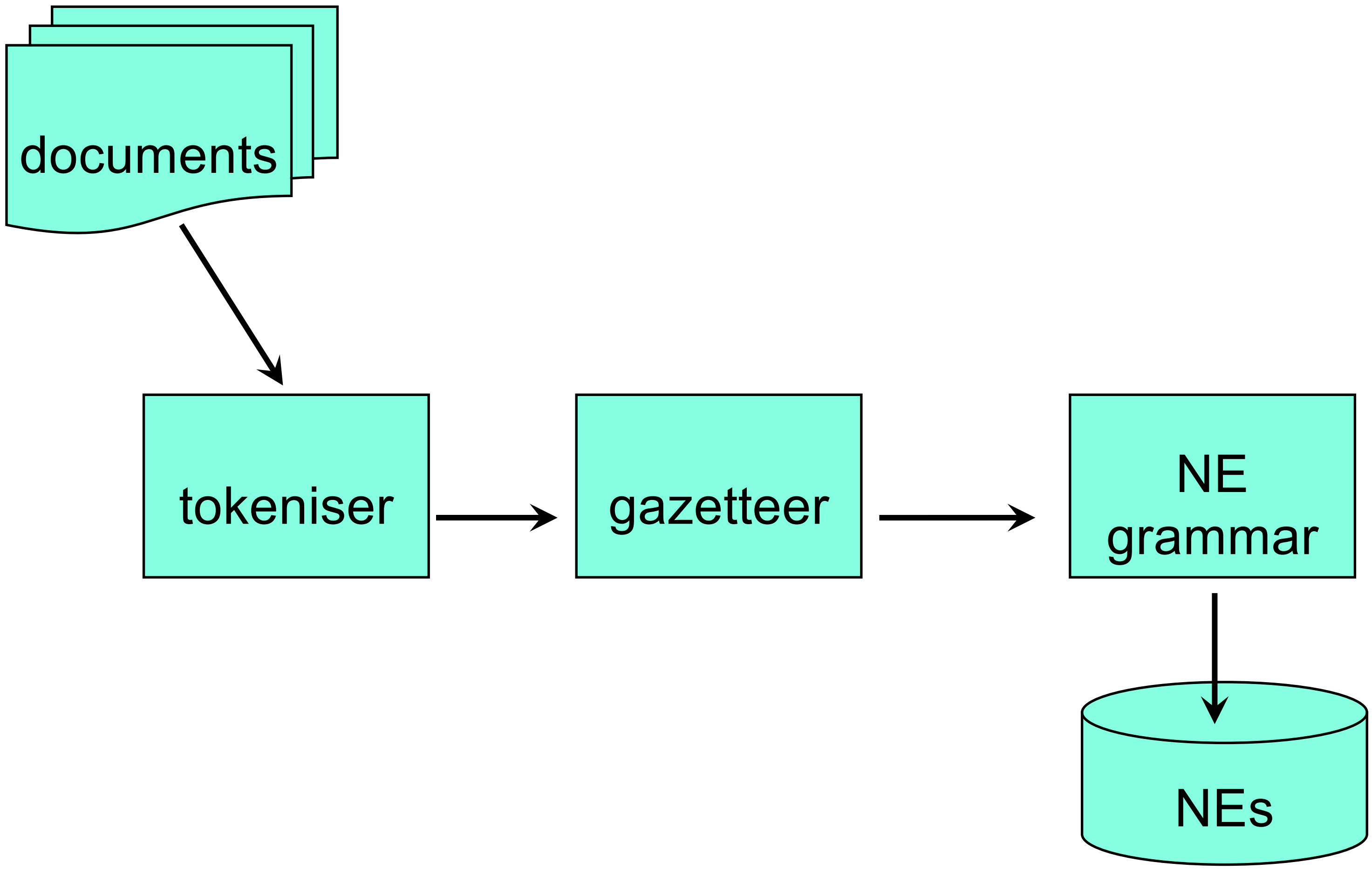
Named Entity Recognition

- 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
- Additional Features
  - Gazetteer
  - Word Capitalize
  - Beginning of the sentence
  - Is it all capitalized

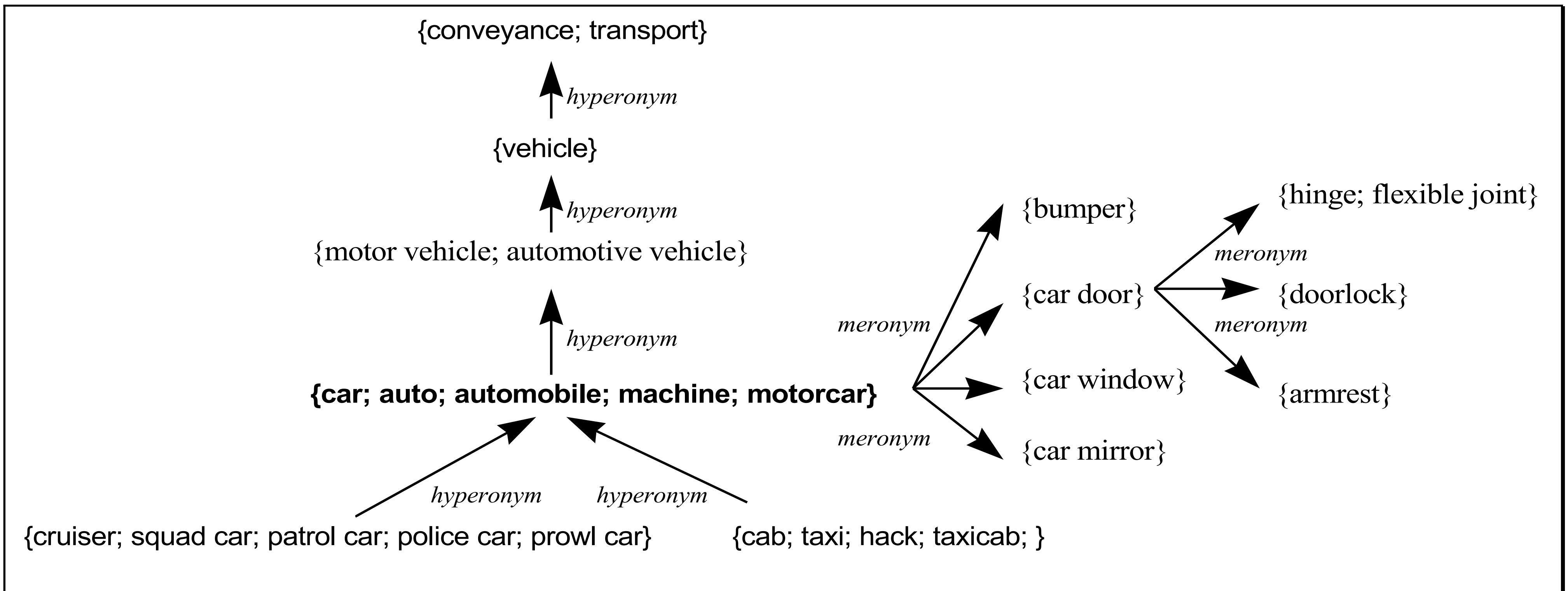
Approach con't

- NE task in two parts:
  - Recognising the entity boundaries
  - Classifying the entities in the NE categories
- Some work is only on one task or the other
- 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

WordNet

- Developed at Princeton by George Miller and his team as a model of the mental lexicon.
- Semantic network in which concepts are defined in terms of relations to other concepts.
- Structure:
  - organized around the notion of synsets (sets of synonymous words)
  - basic semantic relations between these synsets
  - Initially no glosses
  - Main revision after tagging the Brown corpus with word meanings: SemCor.
  - http://www.cogsci.princeton.edu/~wn/w3wn.html

Structure



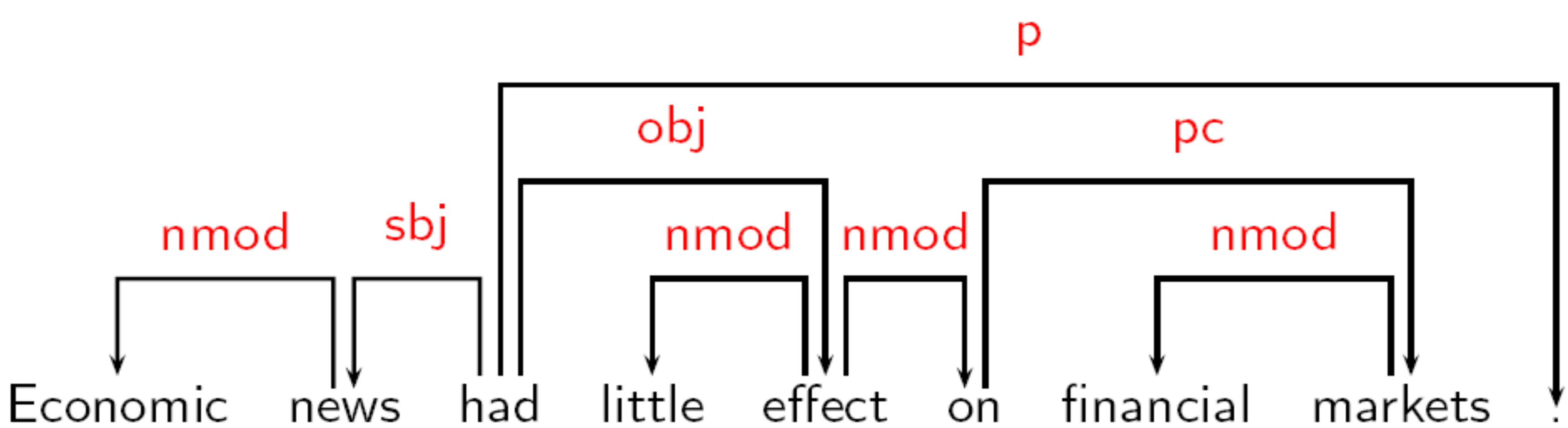
{conveyance; transport}
hyperonym
{vehicle}
hyperonym
{motor vehicle; automotive vehicle}
hyperonym
{car; auto; automobile; machine; motorcar}
hyperonym
hyperonym
{cruiser; squad car; patrol car; police car; prowl car}
{cab; taxi; hack; taxicab; }
meronym
meronym
{bumper}
{car door}
{car window}
{car mirror}
meronym
meronym
{hinge; flexible joint}
{doorlock}
{armrest}

Syntactic Parsing

Dependency Syntax

- The basic idea:
  - Syntactic structure consists of lexical items, linked by binary asymmetric relations called dependencies.

Dependency Structure

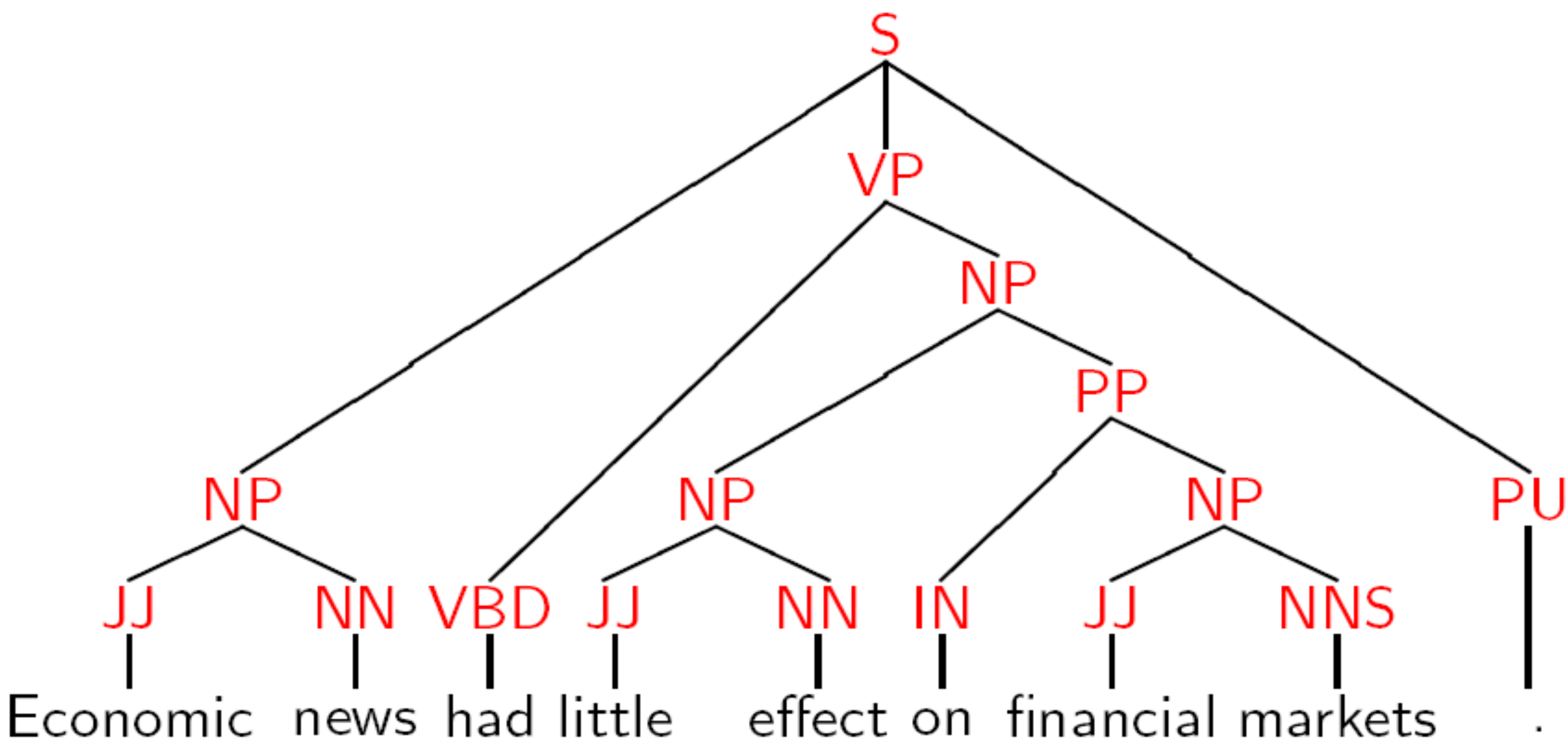


Terminology

Superior	**Inferior**
Head	Dependent
Governor	Modifier
Regent	Subordinate
⋮	⋮

Phrase Structure

(or Constituent Structure)



Comparison

- Dependency structures explicitly represent
  - head-dependent relations (directed arcs),
  - functional categories (arc labels),
  - possibly some structural categories (parts-of-speech).
- Phrase structures explicitly represent
  - phrases (nonterminal nodes),
  - structural categories (nonterminal labels),
  - possibly some functional categories (grammatical functions).
- Hybrid representations may combine all elements.

Predicate Argument Structures

Shallow semantics from predicate argument structures

- In an event:
  - target words describe relation among different entities
  - the participants are often seen as predicate's arguments.
- Example:

  a phosphor gives off electromagnetic energy in this form

Shallow semantics from predicate argument structures

- In an event:
  - target words describe relation among different entities
  - the participants are often seen as predicate's arguments.
- Example:

  [$_{Arg0}$ a phosphor] [$_{predicate}$ gives off] [$_{Arg1}$ electromagnetic energy] [$_{ArgM}$ in this form]

Shallow semantics from predicate argument structures

- In an event:
  - target words describe relation among different entities
  - the participants are often seen as predicate's arguments.
- Example:

  [$_{Arg0}$ a phosphor] [$_{predicate}$ gives off] [$_{Arg1}$ electromagnetic energy] [$_{ArgM}$ in this form]

  [$_{ARGM}$ When] [$_{predicate}$ hit] [$_{Arg0}$ by electrons] [$_{Arg1}$ a phosphor]

Example on Predicate Argument Classification

- In an event:
  - target words describe relation among different entities
  - the participants are often seen as predicate's arguments.
- Example:

Paul gives a talk in Rome

Example on Predicate Argument Classification

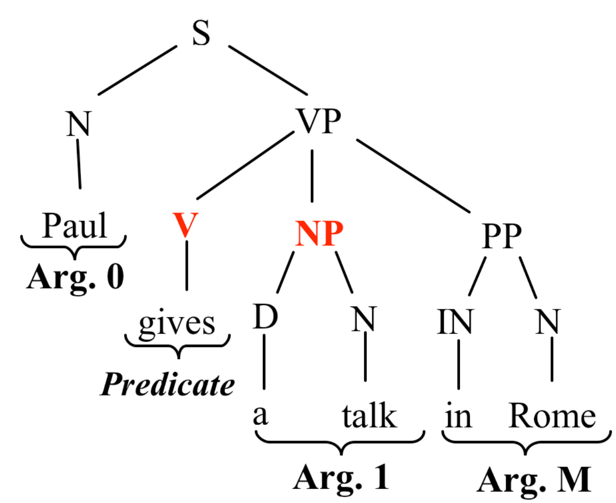
- In an event:
  - target words describe relation among different entities
  - the participants are often seen as predicate's arguments.
- Example:

  [$_{Arg0}$ Paul] [$_{predicate}$ gives] [$_{Arg1}$ a talk] [$_{ArgM}$ in Rome]

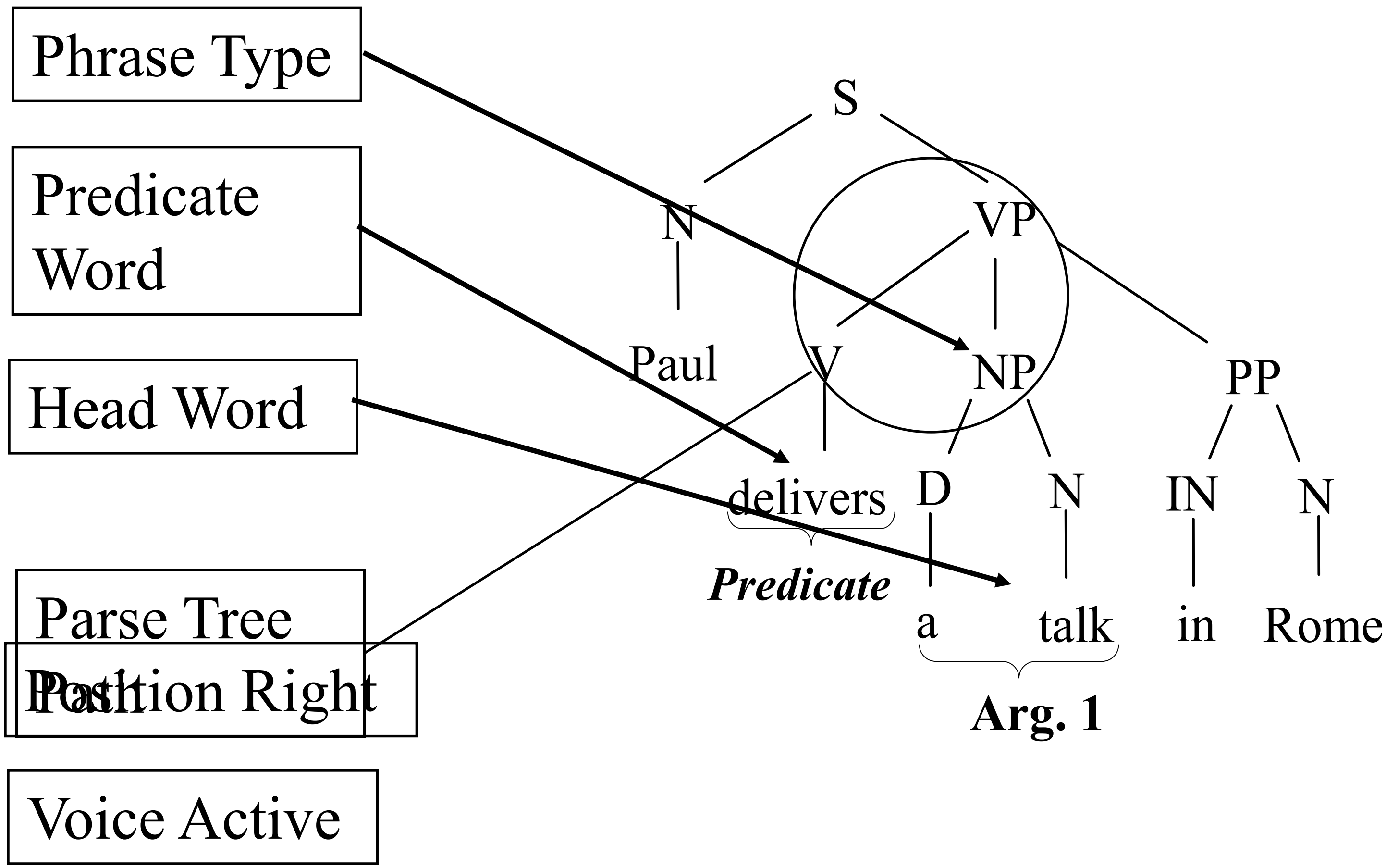
Predicate-Argument Feature Representation

Given a sentence, a predicate *p*:

1. Derive the sentence parse tree
2. For each node pair $<N_p,N_x>$
   a. Extract a feature representation set *F*
   b. If N_x exactly covers the Arg-i, *F* is one of its positive examples
   c. *F* is a negative example otherwise

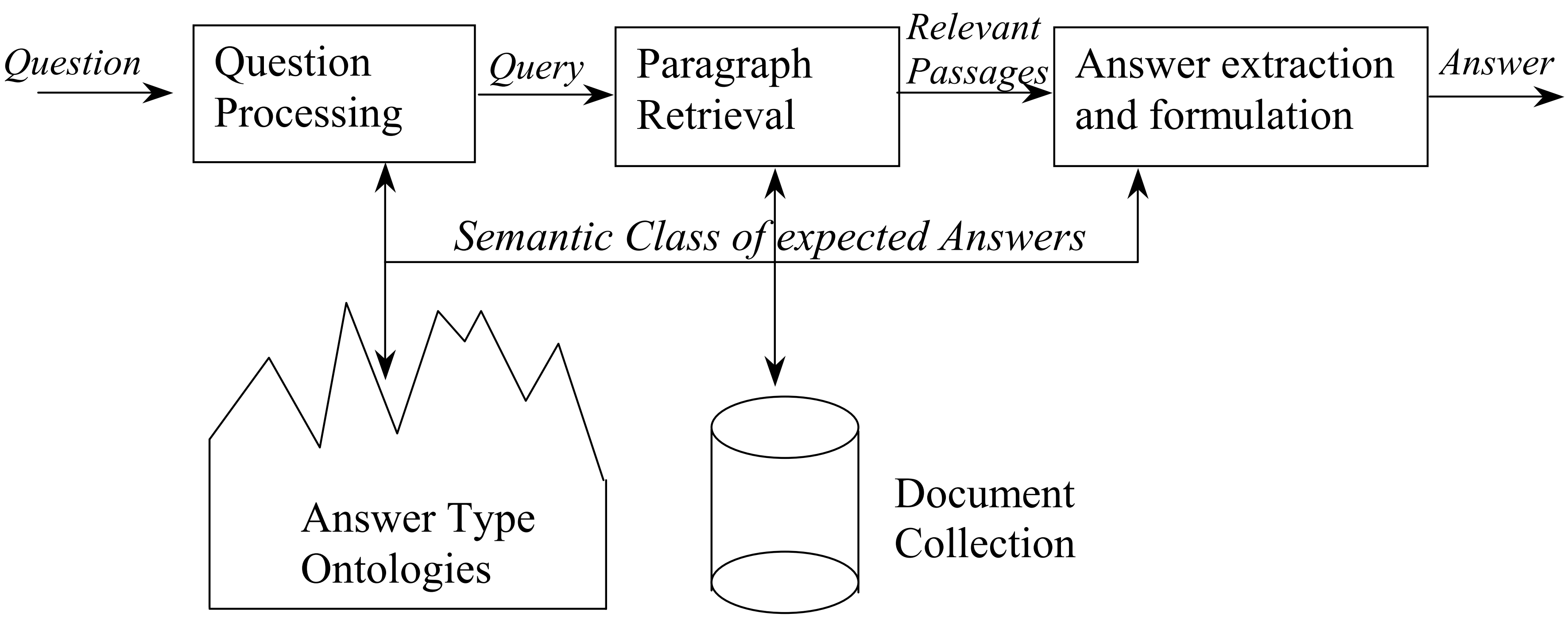


Vector Representation for the linear kernel



Question Answering

Basic Pipeline



Question Classification

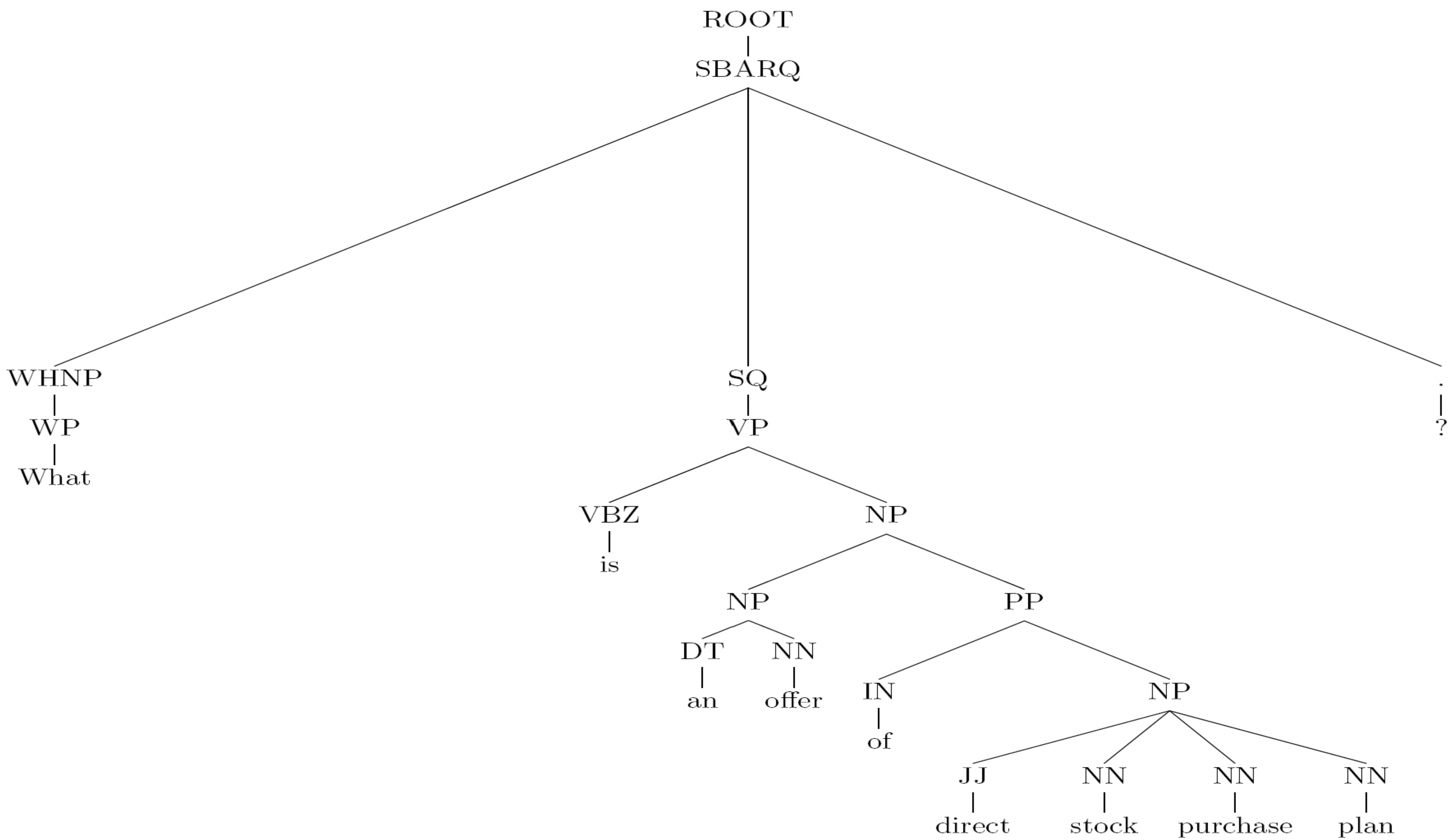
- **Definition**: What does HTML stand for?
- **Description**: What's the final line in the Edgar Allan Poe poem "The Raven"?
- **Entity**: What foods can cause allergic reaction in people?
- **Human**: Who won the Nobel Peace Prize in 1992?
- **Location**: Where is the Statue of Liberty?
- **Manner**: How did Bob Marley die?
- **Numeric**: When was Martin Luther King Jr. born?
- **Organization**: What company makes Bentley cars?

Question Classifier based on Tree Kernels

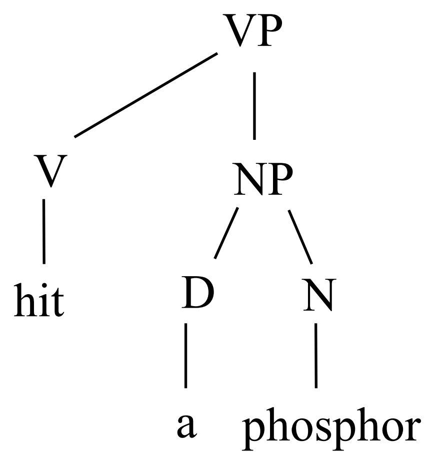
- Question dataset (http://l2r.cs.uiuc.edu/~cogcomp/Data/QA/QC/) [Lin and Roth, 2005])
  - Distributed on 6 categories: Abbreviations, Descriptions, Entity, Human, Location, and Numeric.
- Fixed split 5500 training and 500 test questions
- Using the whole question parse trees
  - Constituent parsing
  - Example

"**What is an offer of direct stock purchase plan ?**"

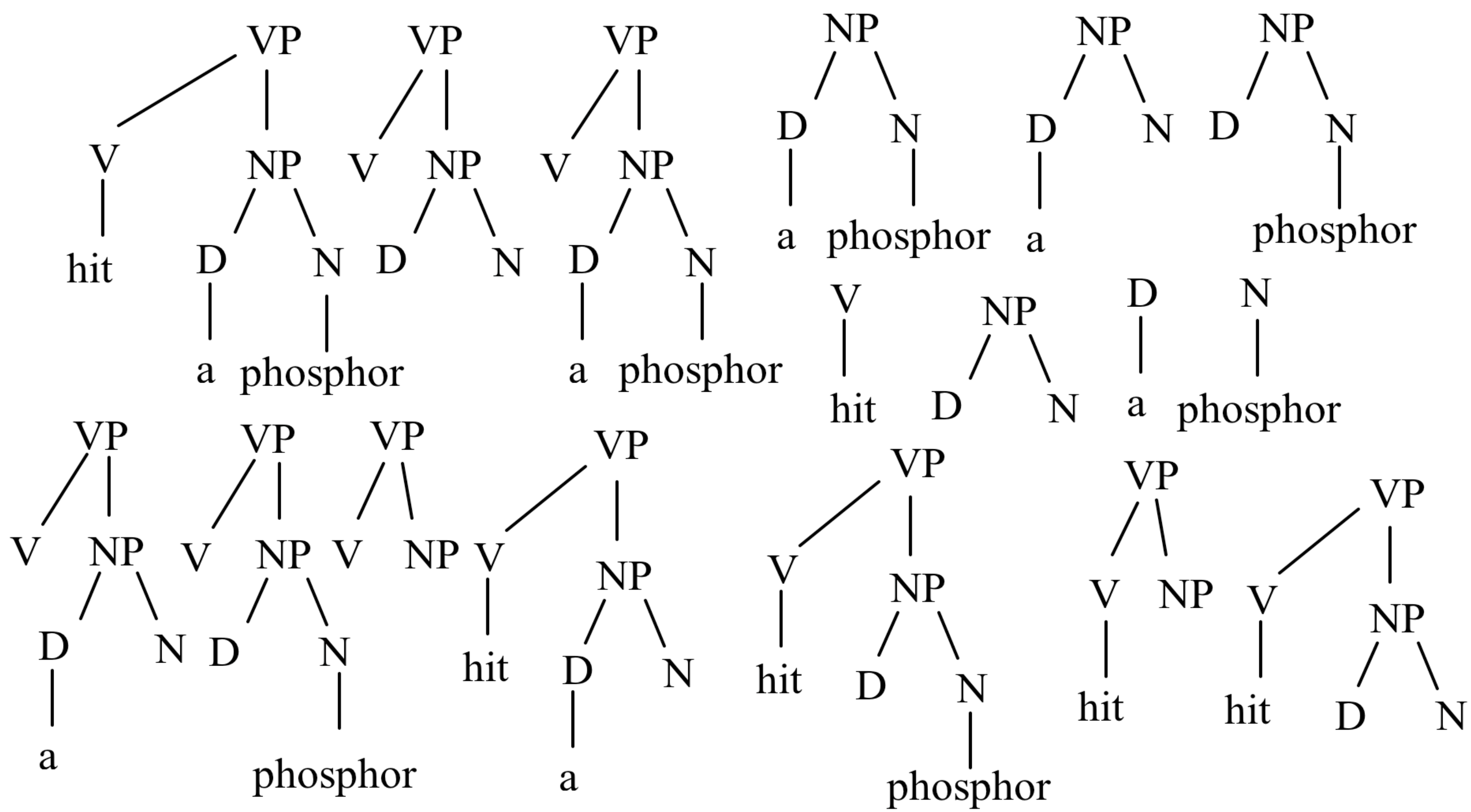
Syntactic Parse Trees (PT)



Similarity based on the number of common substructures

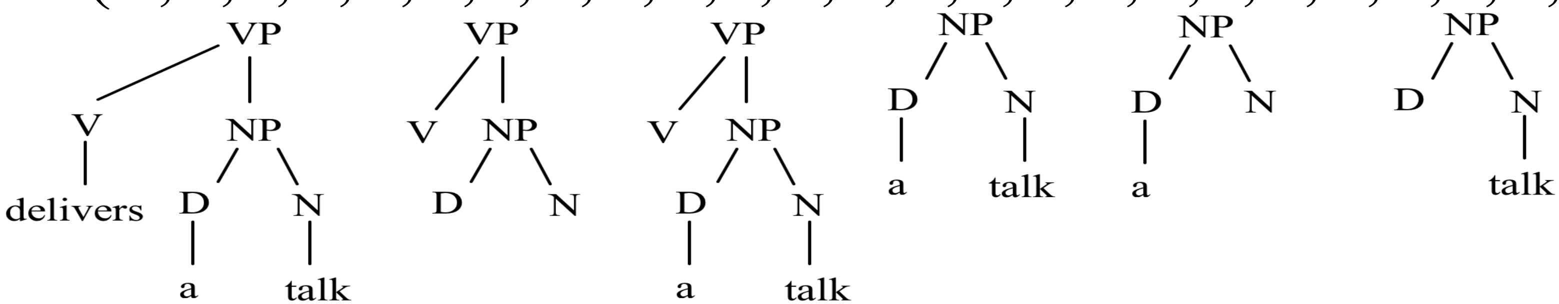


A portion of the substructure set

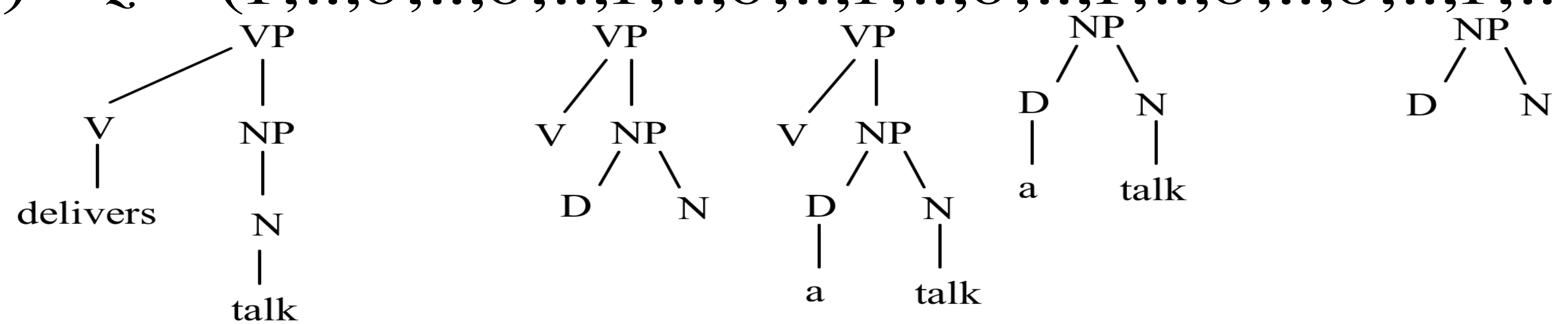


Explicit tree fragment space

$$\phi(T_x) = \vec{x} = (0,..,1,..,0,..,1,..,0,..,1,..,0,..,1,..,0,..,1,..,0,..,1,..,0)$$



$$\phi(T_z) = \vec{z} = (1,..,0,..,0,..,1,..,0,..,1,..,0,..,1,..,0,..,0,..,1,..,0,..,0)$$



- $\vec{x} \cdot \vec{z}$ counts the number of common substructures

Similarity based on WordNet

Inverted Path Length:

$$sim_{IPL}(c_1, c_2) = \frac{1}{(1 + d(c_1, c_2))^{\alpha}}$$

Wu & Palmer:

$$sim_{WUP}(c_1, c_2) = \frac{2\,dep(lso(c_1, c_2))}{d(c_1, lso(c_1, c_2)) + d(c_2, lso(c_1, c_2)) + 2\,dep(lso(c_1, c_2))}$$

Resnik:

$$sim_{RES}(c_1, c_2) = -\log\ P(lso(c_1, c_2))$$

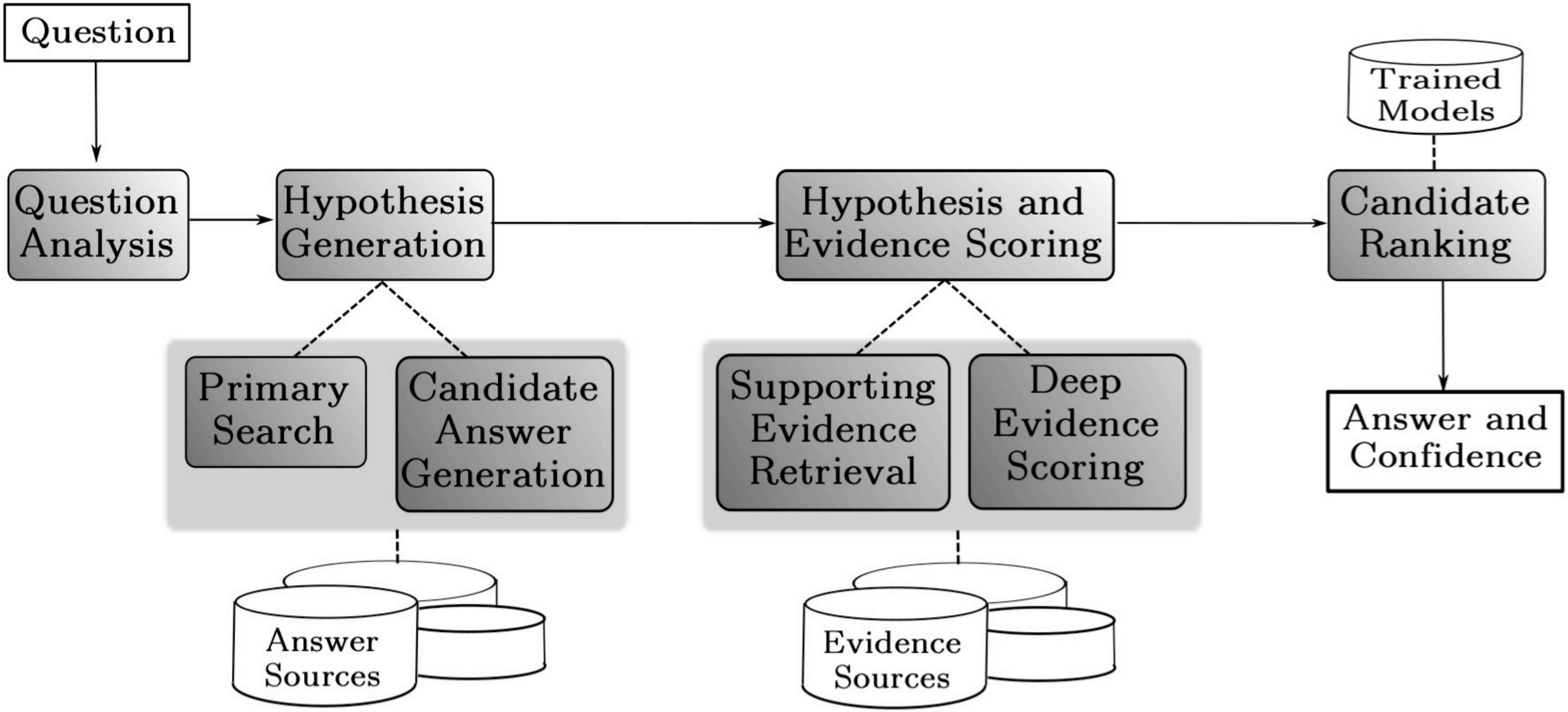
Lin:

$$sim_{LIN}(c_1, c_2) = \frac{2\ \log\ P(lso(c_1, c_2))}{\log\ P(c_1) + \log P\ (c_2)}$$

Question Classification with SSTK

	Accuracy				
λ **parameter**	**0.4**	**0.05**	**0.01**	**0.005**	**0.001**
linear (bow)	0.905				
string matching	0.890	0.910	**0.914**	**0.914**	0.912
full	0.904	**0.924**	0.918	0.922	0.920
full-ic	0.908	**0.922**	0.916	0.918	0.918
path-1	0.906	**0.918**	0.912	**0.918**	0.916
path-2	0.896	0.914	0.914	**0.916**	**0.916**
lin	0.908	**0.924**	0.918	0.922	0.922
wup	0.908	**0.926**	0.918	0.922	0.922

A QA Pipeline: Watson Overview



Thank you

References

- Alessandro Moschitti' handouts http://disi.unitn.eu/~moschitt/teaching.html
- Alessandro Moschitti and Silvia Quarteroni, *Linguistic Kernels for Answer Re-ranking in Question Answering Systems,* Information and Processing Management, ELSEVIER,2010.
- Yashar Mehdad, Alessandro Moschitti and Fabio Massimo Zanzotto. *Syntactic/Semantic Structures for Textual Entailment Recognition*. Human Language Technology - North American chapter of the Association for Computational Linguistics (HLT-NAACL), 2010, Los Angeles, Calfornia.
- Daniele Pighin and Alessandro Moschitti. *On Reverse Feature Engineering of Syntactic Tree Kernels*. In Proceedings of the 2010 Conference on Natural Language Learning, Upsala, Sweden, July 2010. Association for Computational Linguistics.
- Thi Truc Vien Nguyen, Alessandro Moschitti and Giuseppe Riccardi. *Kernel-based Reranking for Entity Extraction.* In proceedings of the 23rd International Conference on Computational Linguistics (COLING), August 2010, Beijing, China.

References

- Alessandro Moschitti. *Syntactic and semantic kernels for short text pair categorization*. In Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009), pages 576–584, Athens, Greece, March 2009.
- Truc-Vien Nguyen, Alessandro Moschitti, and Giuseppe Riccardi. *Convolution kernels on constituent, dependency and sequential structures for relation extraction*. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, pages 1378–1387, Singapore, August 2009.
- Marco Dinarelli, Alessandro Moschitti, and Giuseppe Riccardi. *Re-ranking models based-on small training data for spoken language understanding*. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, pages 1076–1085, Singapore, August 2009.
- Alessandra Giordani and Alessandro Moschitti. *Syntactic Structural Kernels for Natural Language Interfaces to Databases*. In ECML/PKDD, pages 391–406, Bled, Slovenia, 2009.

References

- Alessandro Moschitti, Daniele Pighin and Roberto Basili*. Tree Kernels for Semantic Role Labeling,* Special Issue on Semantic Role Labeling, Computational Linguistics Journal. March 2008.
- Fabio Massimo Zanzotto, Marco Pennacchiotti and Alessandro Moschitti*, A Machine Learning Approach to Textual Entailment Recognition,* Special Issue on Textual Entailment Recognition, Natural Language Engineering, Cambridge University Press., 2008
- Mona Diab, Alessandro Moschitti, Daniele Pighin, *Semantic Role Labeling Systems for Arabic Language using Kernel Methods*. In proceedings of the 46th Conference of the Association for Computational Linguistics (ACL'08). Main Paper Section. Columbus, OH, USA, June 2008.
- Alessandro Moschitti, Silvia Quarteroni, Kernels on Linguistic Structures for Answer Extraction. In proceedings of the 46th Conference of the Association for Computational Linguistics (ACL'08). Short Paper Section. Columbus, OH, USA, June 2008.

References

- Yannick Versley, Simone Ponzetto, Massimo Poesio, Vladimir Eidelman, Alan Jern, Jason Smith, Xiaofeng Yang and Alessandro Moschitti, *BART: A Modular Toolkit for Coreference Resolution*, In Proceedings of the Conference on Language Resources and Evaluation, Marrakech, Marocco, 2008.
- Alessandro Moschitti, *Kernel Methods, Syntax and Semantics for Relational Text Categorization*. In proceeding of ACM 17th Conference on Information and Knowledge Management (CIKM). Napa Valley, California, 2008.
- Bonaventura Coppola, Alessandro Moschitti, and Giuseppe Riccardi. *Shallow semantic parsing for spoken language understanding*. In Proceedings of HLT-NAACL Short Papers, pages 85–88, Boulder, Colorado, June 2009. Association for Computational Linguistics.
- Alessandro Moschitti and Fabio Massimo Zanzotto, *Fast and Effective Kernels for Relational Learning from Texts*, Proceedings of The 24th Annual International Conference on Machine Learning (ICML 2007).

References

- Alessandro Moschitti, Silvia Quarteroni, Roberto Basili and Suresh Manandhar, *Exploiting Syntactic and Shallow Semantic Kernels for Question/Answer Classification*, Proceedings of the 45th Conference of the Association for Computational Linguistics (ACL), Prague, June 2007.
- Alessandro Moschitti and Fabio Massimo Zanzotto, *Fast and Effective Kernels for Relational Learning from Texts*, Proceedings of The 24th Annual International Conference on Machine Learning (ICML 2007), Corvallis, OR, USA.
- Daniele Pighin, Alessandro Moschitti and Roberto Basili, *RTV: Tree Kernels for Thematic Role Classification*, Proceedings of the 4th International Workshop on Semantic Evaluation (SemEval-4), English Semantic Labeling, Prague, June 2007.
- Stephan Bloehdorn and Alessandro Moschitti, *Combined Syntactic and Semanitc Kernels for Text Classification*, to appear in the 29th European Conference on Information Retrieval (ECIR), April 2007, Rome, Italy.
- Fabio Aiolli, Giovanni Da San Martino, Alessandro Sperduti, and Alessandro Moschitti, *Efficient Kernel-based Learning for Trees*, to appear in the IEEE Symposium on Computational Intelligence and Data Mining (CIDM), Honolulu, Hawaii, 2007

References

- Alessandro Moschitti, Silvia Quarteroni, Roberto Basili and Suresh Manandhar, *Exploiting Syntactic and Shallow Semantic Kernels for Question/Answer Classification*, Proceedings of the 45th Conference of the Association for Computational Linguistics (ACL), Prague, June 2007.
- Alessandro Moschitti, Giuseppe Riccardi, Christian Raymond, *Spoken Language Understanding with Kernels for Syntactic/Semantic Structures*, Proceedings of IEEE Automatic Speech Recognition and Understanding Workshop (ASRU2007), Kyoto, Japan, December 2007
- Stephan Bloehdorn and Alessandro Moschitti, *Combined Syntactic and Semantic Kernels for Text Classification*, to appear in the 29th European Conference on Information Retrieval (ECIR), April 2007, Rome, Italy.
- Stephan Bloehdorn, Alessandro Moschitti: Structure and semantics for expressive text kernels. In proceeding of ACM 16th Conference on Information and Knowledge Management (CIKM-short paper) 2007: 861-864, Portugal.

References

- Fabio Aiolli, Giovanni Da San Martino, Alessandro Sperduti, and Alessandro Moschitti, *Efficient Kernel-based Learning for Trees*, to appear in the IEEE Symposium on Computational Intelligence and Data Mining (CIDM), Honolulu, Hawaii, 2007.
- Alessandro Moschitti*, Efficient Convolution Kernels for Dependency and Constituent Syntactic Trees.* In Proceedings of the 17th European Conference on Machine Learning, Berlin, Germany, 2006.
- Fabio Aiolli, Giovanni Da San Martino, Alessandro Sperduti, and Alessandro Moschitti, *Fast On-line Kernel Learning for Trees*, International Conference on Data Mining (ICDM) 2006 (short paper).
- Stephan Bloehdorn, Roberto Basili, Marco Cammisa, Alessandro Moschitti, *Semantic Kernels for Text Classification based on Topological Measures of Feature Similarity*. In Proceedings of the 6th IEEE International Conference on Data Mining (ICDM 06), Hong Kong, 18-22 December 2006. (short paper).

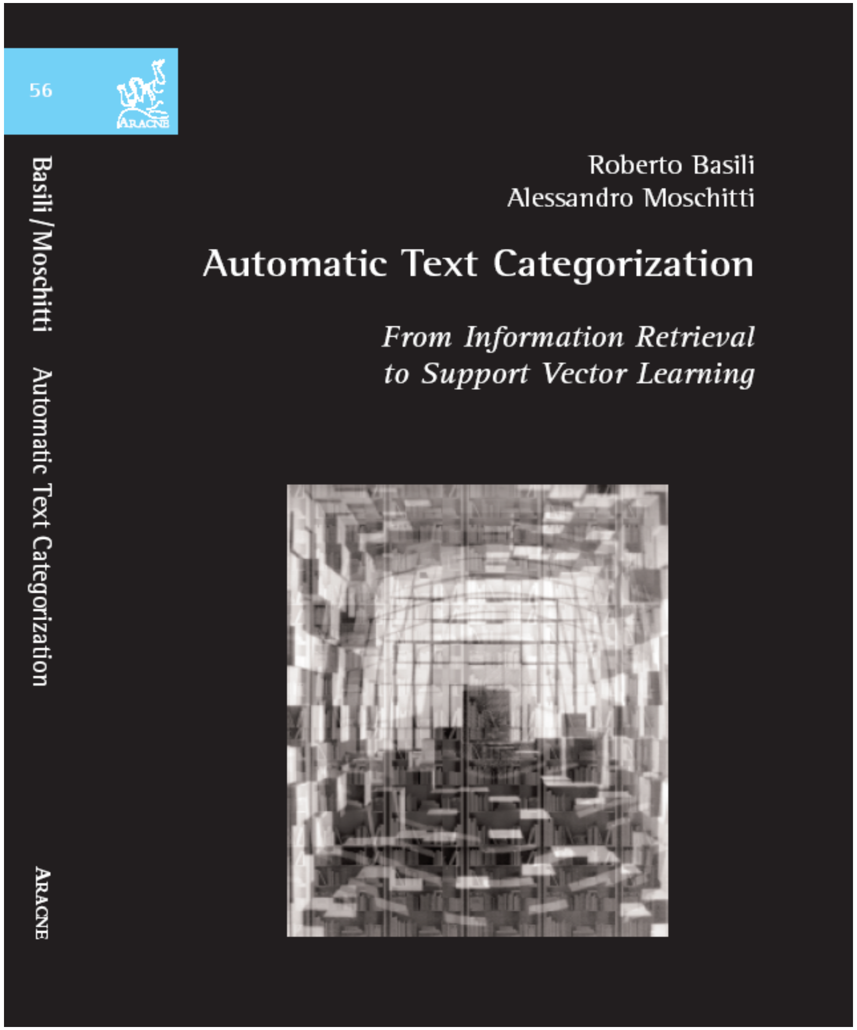
References

- Roberto Basili, Marco Cammisa and Alessandro Moschitti, *A Semantic Kernel to classify texts with very few training examples*, in Informatica, an international journal of Computing and Informatics, 2006.
- Fabio Massimo Zanzotto and Alessandro Moschitti, *Automatic learning of textual entailments with cross-pair similarities*. In Proceedings of COLING-ACL, Sydney, Australia, 2006.
- Ana-Maria Giuglea and Alessandro Moschitti*, Semantic Role Labeling via FrameNet, VerbNet and PropBank.* In Proceedings of COLING-ACL, Sydney, Australia, 2006.
- Alessandro Moschitti*, Making tree kernels practical for natural language learning.* In Proceedings of the Eleventh International Conference on European Association for Computational Linguistics, Trento, Italy, 2006.
- Alessandro Moschitti, Daniele Pighin and Roberto Basili. *Semantic Role Labeling via Tree Kernel joint inference*. In Proceedings of the 10th Conference on Computational Natural Language Learning, New York, USA, 2006.

References

- Roberto Basili, Marco Cammisa and Alessandro Moschitti, *Effective use of Wordnet semantics via kernel-based learning*. In Proceedings of the 9th Conference on Computational Natural Language Learning (CoNLL 2005), Ann Arbor (MI), USA, 2005
- Alessandro Moschitti, *A study on Convolution Kernel for Shallow Semantic Parsing*. In proceedings of the 42-th Conference on Association for Computational Linguistic (ACL-2004), Barcelona, Spain, 2004.
- Alessandro Moschitti and Cosmin Adrian Bejan, *A Semantic Kernel for Predicate Argument Classification.* In proceedings of the Eighth Conference on Computational Natural Language Learning (CoNLL-2004), Boston, MA, USA, 2004.

An introductory book on SVMs, Kernel methods and Text Categorization



Non-exhaustive reference list from other authors

- V. Vapnik. The Nature of Statistical Learning Theory. Springer, 1995.
- P. Bartlett and J. Shawe-Taylor, 1998. Advances in Kernel Methods - Support Vector Learning, chapter Generalization Performance of Support Vector Machines and other Pattern Classifiers. MIT Press.
- David Haussler. 1999. Convolution kernels on discrete structures. Technical report, Dept. of Computer Science, University of California at Santa Cruz.
- Lodhi, Huma, Craig Saunders, John Shawe Taylor, Nello Cristianini, and Chris Watkins. Text classification using string kernels. JMLR,2000
- Schölkopf, Bernhard and Alexander J. Smola. 2001. Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press, Cambridge, MA, USA.

Non-exhaustive reference list from other authors

- N. Cristianini and J. Shawe-Taylor, *An introduction to support vector machines (and other kernel-based learning methods)* Cambridge University Press, 2002
- M. Collins and N. Duffy, New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In ACL02, 2002.
- Hisashi Kashima and Teruo Koyanagi. 2002. Kernels for semi-structured data. In Proceedings of ICML’02.
- S.V.N. Vishwanathan and A.J. Smola. Fast kernels on strings and trees. In Proceedings of NIPS, 2002.
- Nicola Cancedda, Eric Gaussier, Cyril Goutte, and Jean Michel Renders. 2003. Word sequence kernels. Journal of Machine Learning Research, 3:1059–1082. D. Zelenko, C. Aone, and A. Richardella. Kernel methods for relation extraction. JMLR, 3:1083–1106, 2003.

Non-exhaustive reference list from other authors

- Taku Kudo and Yuji Matsumoto. 2003. Fast methods for kernel-based text analysis. In Proceedings of ACL'03.
- Dell Zhang and Wee Sun Lee. 2003. Question classification using support vector machines. In Proceedings of SIGIR'03, pages 26–32.
- Libin Shen, Anoop Sarkar, and Aravind k. Joshi. Using LTAG Based Features in Parse Reranking. In Proceedings of EMNLP'03, 2003
- C. Cumby and D. Roth. Kernel Methods for Relational Learning. In Proceedings of ICML 2003, pages 107–114, Washington, DC, USA, 2003.
- J. Shawe-Taylor and N. Cristianini. Kernel Methods for Pattern Analysis. Cambridge University Press, 2004.
- A. Culotta and J. Sorensen. Dependency tree kernels for relation extraction. In Proceedings of the 42nd Annual Meeting on ACL, Barcelona, Spain, 2004.

Non-exhaustive reference list from other authors

- Kristina Toutanova, Penka Markova, and Christopher Manning. The Leaf Path Projection View of Parse Trees: Exploring String Kernels for HPSG Parse Selection. In Proceedings of EMNLP 2004.
- Jun Suzuki and Hideki Isozaki. 2005. Sequence and Tree Kernels with Statistical Feature Mining. In Proceedings of NIPS'05.
- Taku Kudo, Jun Suzuki, and Hideki Isozaki. 2005. Boosting based parse reranking with subtree features. In Proceedings of ACL'05.
- R. C. Bunescu and R. J. Mooney. Subsequence kernels for relation extraction. In Proceedings of NIPS, 2005.
- R. C. Bunescu and R. J. Mooney. A shortest path dependency kernel for relation extraction. In Proceedings of EMNLP, pages 724–731, 2005.
- S. Zhao and R. Grishman. Extracting relations with integrated information using kernel methods. In Proceedings of the 43rd Meeting of the ACL, pages 419–426, Ann Arbor, Michigan, USA, 2005.

Non-exhaustive reference list from other authors

- J. Kazama and K. Torisawa. Speeding up Training with Tree Kernels for Node Relation Labeling. In Proceedings of EMNLP 2005, pages 137–144, Toronto, Canada, 2005.
- M. Zhang, J. Zhang, J. Su, , and G. Zhou. A composite kernel to extract relations between entities with both flat and structured features. In Proceedings of COLING-ACL 2006, pages 825–832, 2006.
- M. Zhang, G. Zhou, and A. Aw. Exploring syntactic structured features over parse trees for relation extraction using kernel methods. Information Processing and Management, 44(2):825–832, 2006.
- G. Zhou, M. Zhang, D. Ji, and Q. Zhu. Tree kernel-based relation extraction with context-sensitive structured parse tree information. In Proceedings of EMNLP-CoNLL 2007, pages 728–736, 2007.

Non-exhaustive reference list from other authors

- Ivan Titov and James Henderson. Porting statistical parsers with data-defined kernels. In Proceedings of CoNLL-X, 2006
- Min Zhang, Jie Zhang, and Jian Su. 2006. Exploring Syntactic Features for Relation Extraction using a Convolution tree kernel. In Proceedings of NAACL.
- M. Wang. A re-examination of dependency path kernels for relation extraction. In Proceedings of the 3rd International Joint Conference on Natural Language Processing-IJCNLP, 2008.