Machine Learning for Natural Language Processing

Traian Rebedea, Ștefan Rușeți

traian.rebedea@cs.pub.ro, stefan.ruseti@cs.pub.ro

LeMAS 2016 – Summer School

Outline

- Intro
- Basic notions of Machine Learning (ML)
- Basic notions in Natural Language Processing (NLP)
- Sample problems and solutions
    - Finding vandalism on Wikipedia
    - Sexual predator detection
    - Question-answering over linked data (Stefan)
- Practical ML
    - R
    - Python (sklearn)
    - Java (Weka)
    - Others
- Practical NLP
    - Java (CoreNLP)
    - Python (nltk)
    - Cython (spaCy)
    - Others

About me – Traian

Started working in 2003 (year 2 of studies): J2EE developer

Then founded my own company with a colleague (in 2005)

- Web projects, involved in some startups as tech advisors (e.g. Happyfish TV)

Started teaching at A&C, UPB (in 2006)

- TA for Algorithms, Natural Language Processing

Soon I also started my PhD (in 2007)

- Natural Language Processing, Discourse Analysis, Technology-Enhanced Learning

Now I am lecturer for: Algorithm Design, Algorithm Design and Complexity, Symbolic and Statistical Learning, Information Retrieval

Working on several topics in NLP: opinion mining, conversational agents, question-answering, culturonomics

Interested in all new projects that use NLP, ML and IR as well as any other “smart” applications

NLP and ML-related collaborations with PeopleGraph, Teamnet, Bitdefender, Treeworks

Why machine learning?

"Simply put, **machine learning is the part of artificial intelligence that actually works**" (Forbes, link)

Most popular course on Coursera

Popular?

- Meaning practical: ML results are nice and easy to show to others
- Meaning well-paid/in demand: companies are increasing demand for ML specialists

Large volumes of data, corpora, etc.

Computer science, data science, almost any other domain (especially in research)

Why ML?

Mix of Computer science and Math (Statistics) skills

Why is math essential to ML?

http://courses.washington.edu/css490/2012.Winter/lecture_slides/02_math_essentials.pdf

"To get really useful results, you need good mathematical intuitions about certain general machine learning principles, as well as the **inner workings of the individual algorithms**."

Basic notions for ML

- Different tasks in ML
- Supervised
  - Regression
  - Classification
- Unsupervised
  - Clustering
  - Dimensionality reduction / visualization
- Semi-supervised
  - Label propagation
- Advanced notions are not treated here (reinforcement, deep learning, etc.)

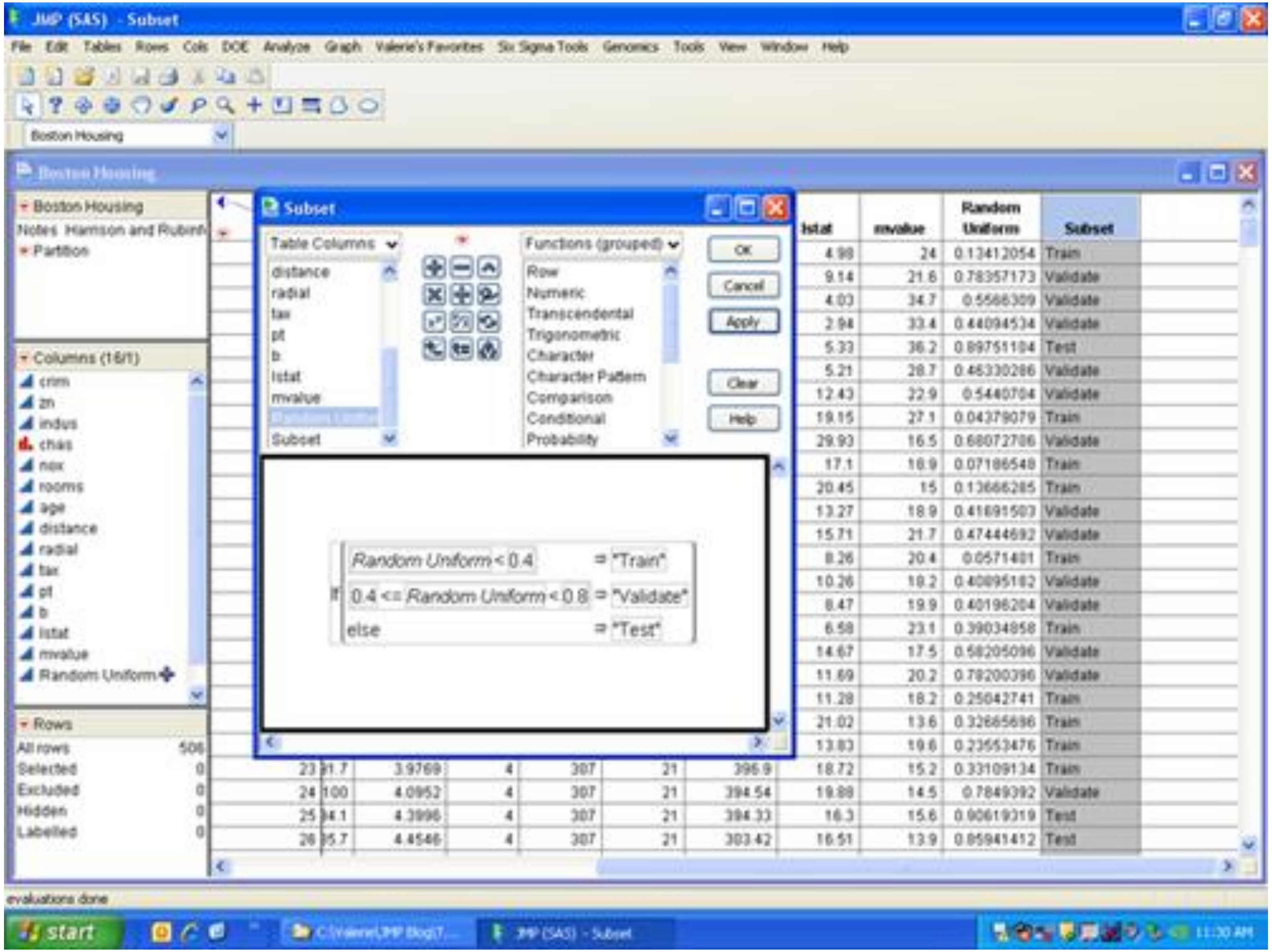
Supervised learning

- Dataset consisting of **labeled** examples
- Each example has one or several attributes and a dependent variable (label, output, response, predicted variable)
- After building, selection and assessing the model on the labeled data, it is used to find labels for new data

Supervised learning

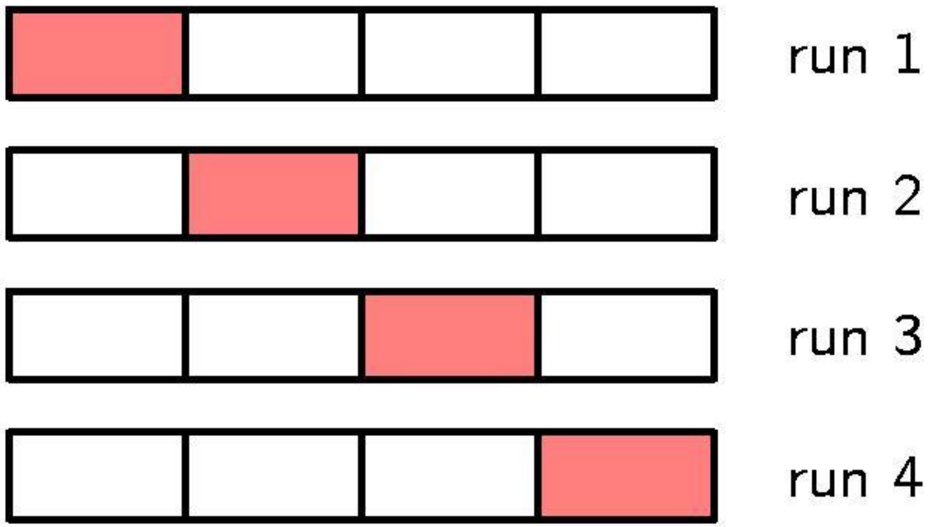
- Dataset is usually split into three parts
    - A rule is 50-25-25

- Training set
    - Used to **build models**
    - Compute the basic parameters of the model
- Holdout/validation set
    - Used to **find out the best model** and **adjust some parameters** of the model (usually, in order to minimize some error)
    - Also called **model selection**
- Test set
    - Used to **assess the final model on new data**, after model building and selection



- Source: http://blogs.sas.com/content/jmp/2010/07/06/train-validate-and-test-for-data-mining-in-jmp/

Cross-validation



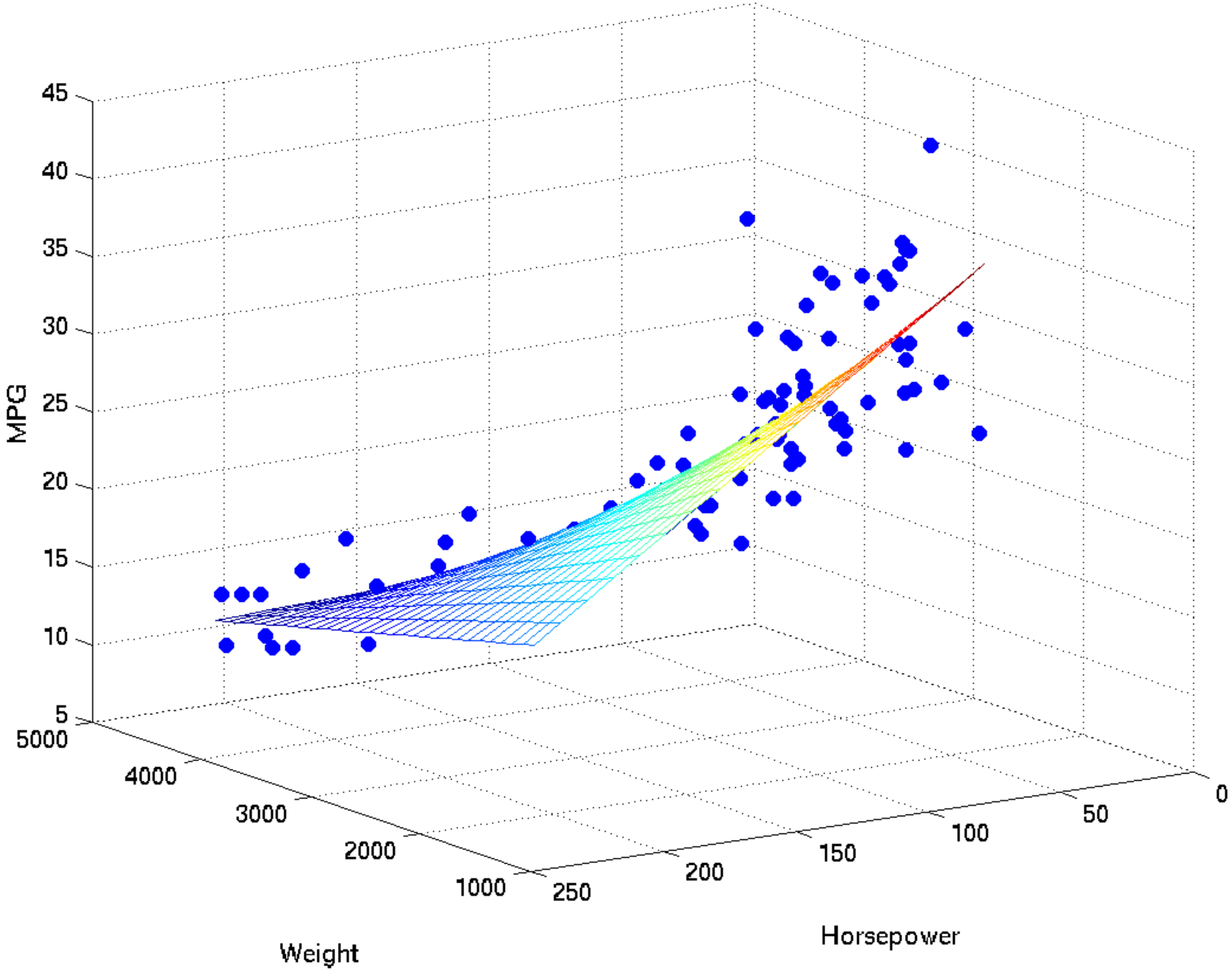
Main disadvantages:

- no of runs is increased by a factor of S
- Multiple parameters for the same model for different runs

If no of partitions (S) = no of training instances → leave-one-out (only for small datasets)

Regression

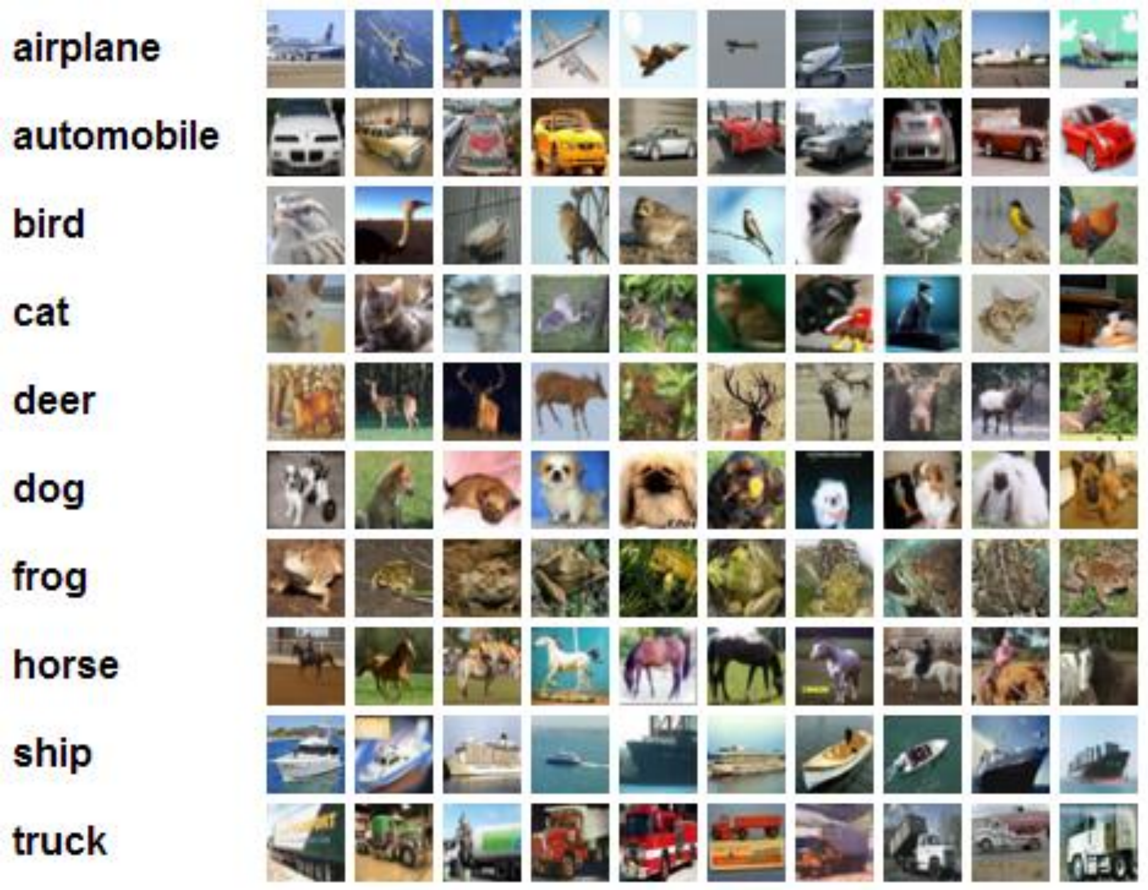
- Supervised learning when the output variable is continous
  - This is the usual textbook definition
  - Also may be used for binary output variables (e.g. Not-spam=0, Spam=1) or output variables that have an ordering (e.g. integer numbers, sentiment levels, etc.)



- Source: http://smlv.cc.gatech.edu/2010/10/06/linear-regression-and-least-squares-estimation/

Classification

- Supervised learning when the output variable is discrete (classes)
  - Can also be used for binary or integer outputs
  - Even if they are ordered, but classification usually does not account for the order



- CIFAR dataset: http://www.cs.toronto.edu/~kriz/cifar.html (also image source)

Classification

- Several important types
  - Binary vs. multiple classes
  - Hard vs. soft
  - Single-label vs. multi-label (per example)
  - Balanced vs. unbalanced classes

- Extensions that are semi-supervised
  - One-class classification
  - PU (Positive and Unlabeled) learning

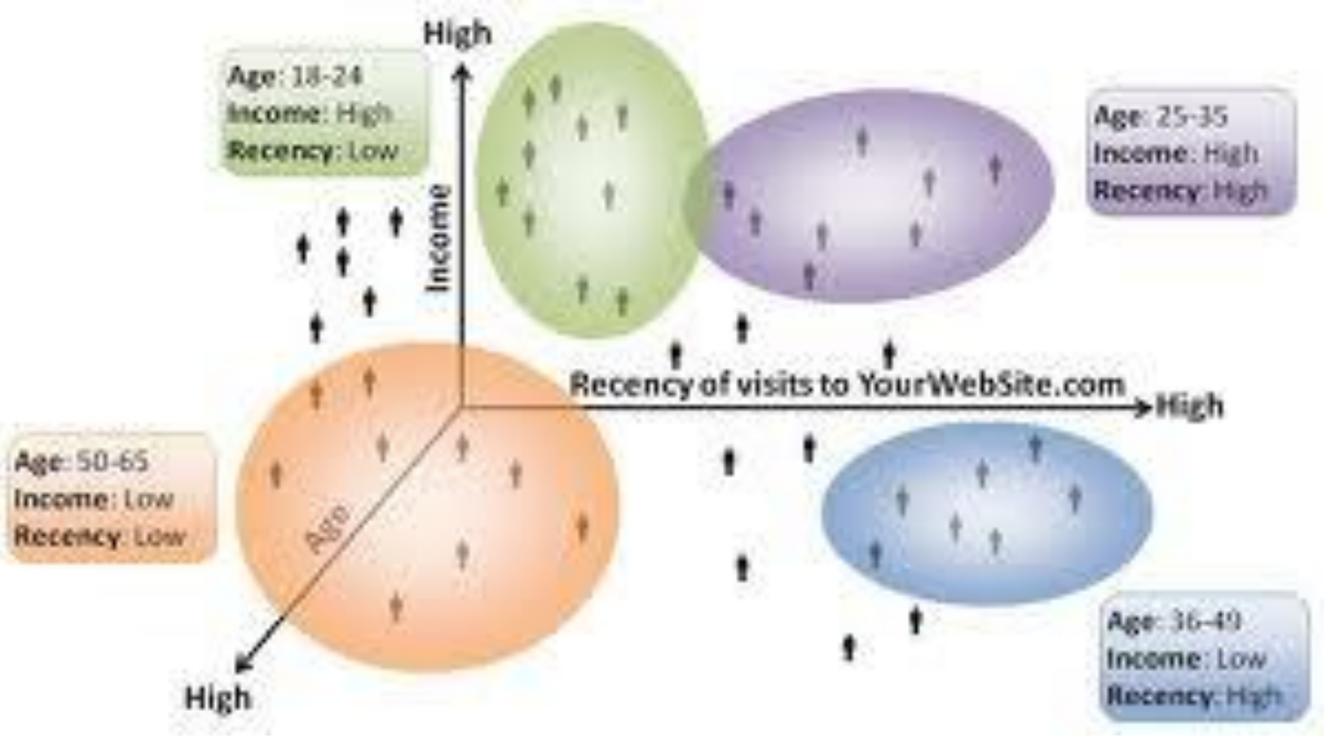
Unsupervised learning

- For an unlabeled dataset, infer properties (find “hidden” structure) about the distribution of the objects in the dataset
  - without any help related to correct answers

- Generally, much more data than for supervised learning
- There are several techniques that can be applied
  - Clustering (cluster analysis)
  - Dimensionality reduction (visualization of data)
  - Association rules, frequent itemsets

Clustering

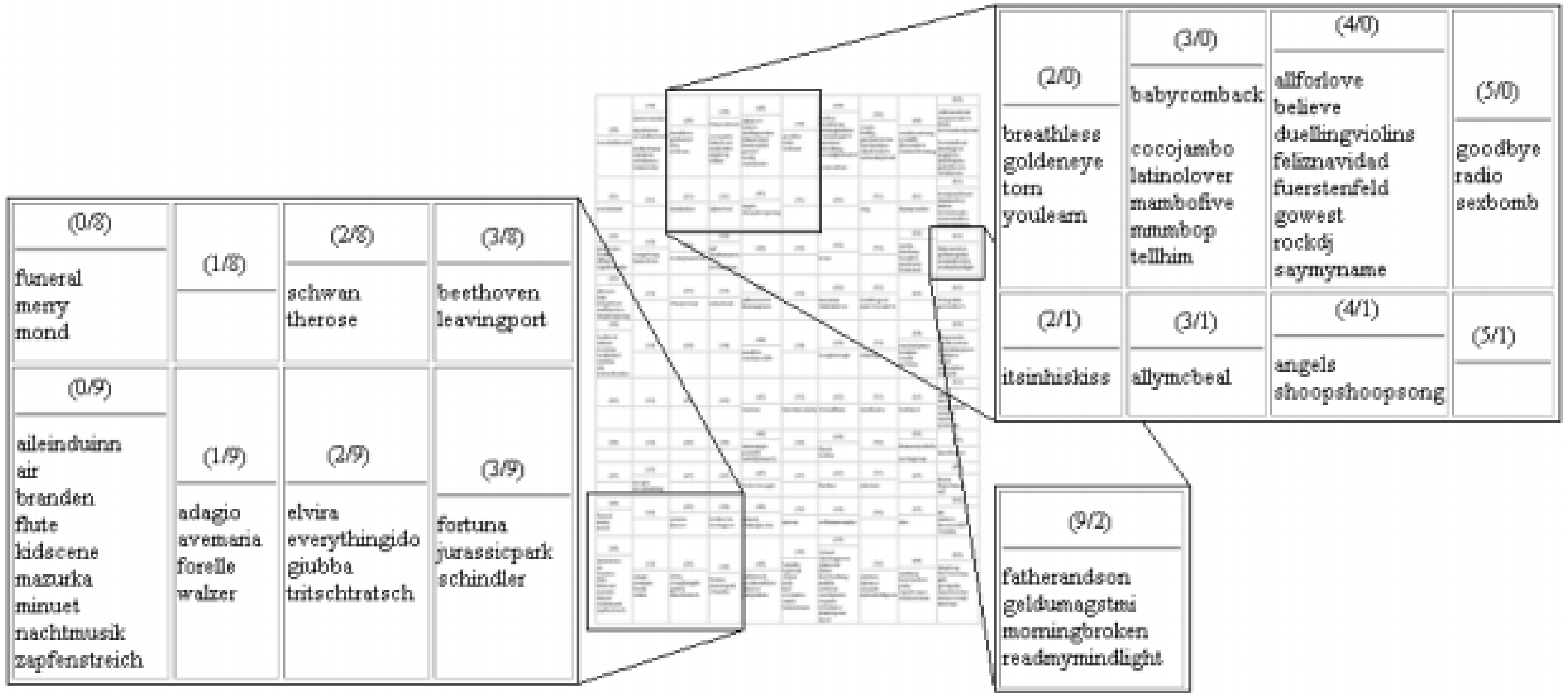
- Partition dataset into clusters (groups) of objects such that the ones in the same cluster are **more similar** to each other than to objects in different clusters
- Similarity
  - The most important notion when clustering
  - “Inverse of a distance”
- Possible objective: approximate the modes of the input dataset distribution



- Source: https://sites.google.com/site/statsr4us/advanced/multivariate/cluster

Dimensionality reduction

- Usually, unsupervised datasets are large both in number of examples and in number of attributes (features)
- Reduce the number of features in order to improve (human) readability and interpretation of data
- Mainly linked to visualization
  - Also used for feature selection, data compression or denoising



- Source: http://www.ifs.tuwien.ac.at/ifs/research/pub_pdf/rau_ecdl01.pdf

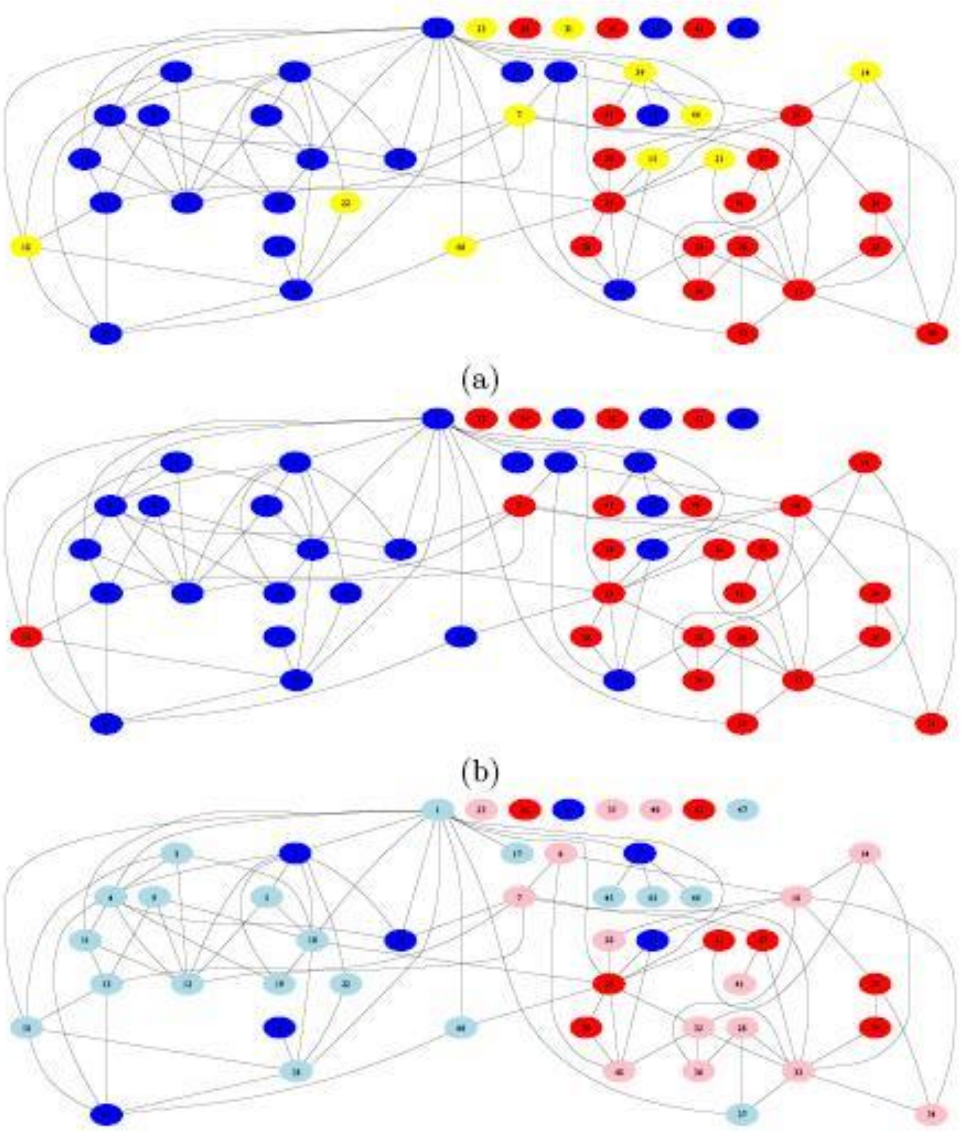
Semi-supervised learning

- Mix of labeled and unlabeled data in the dataset
- Enlarge the labeled dataset with unlabeled instances for which we might determine the correct label with a high probability

- Unlabeled data is much cheaper or simpler to get than labeled data
  - Human annotation either requires experts or is not challenging (it is boring)
  - Annotation may also require specific software or other types of devices
  - Students (or Amazon Mturk users) are not always good annotators
- More information: http://pages.cs.wisc.edu/~jerryzhu/pub/sslicml07.pdf

Label propagation

- Start with the labeled dataset
- Want to assign labels to some (or all) of the unlabeled instances
- Assumption: “**data points that are close have similar labels**”
- Build a fully connected graph with both labeled and unlabeled instances as vertices
- The weight of an edge represents the similarity between the points (or inverse of a distance)
- Repeat until convergence
  - Propagate labels through edges (larger weights allow a more quick propagation)
  - Nodes will have soft labels
- More info: http://lvk.cs.msu.su/~bruzz/articles/classification/zhu02learning.pdf



(a)

(b)

(c)

- Source: http://web.ornl.gov/sci/knowledgediscovery/Projects.htm

Basic Notions of ML

- **How to asses the performance of a model?**
- Used to choose the best model

- Supervised learning
  - Assessing errors of the model on the validation and test sets

Performance measures - supervised

- Regression
  - Mean error, mean squared error (MSE), root MSE, etc.
- Classification
  - Accuracy
  - Precision / recall (possibly weighted)
  - F-measure
  - Confusion matrix
  - TP, TN, FP, FN
  - AUC / ROC

Mean Squared Eror - 2

$$MSE = \frac{1}{n} \sum_{i=1}^{n} \left(y_i - \hat{y}_i\right)^2 = \frac{1}{n} \sum_{i=1}^{n} e_i^2$$

$$MSE = s_e^2 + \overline{e}^2$$

_ _

$$MSE = VARE + MES$$

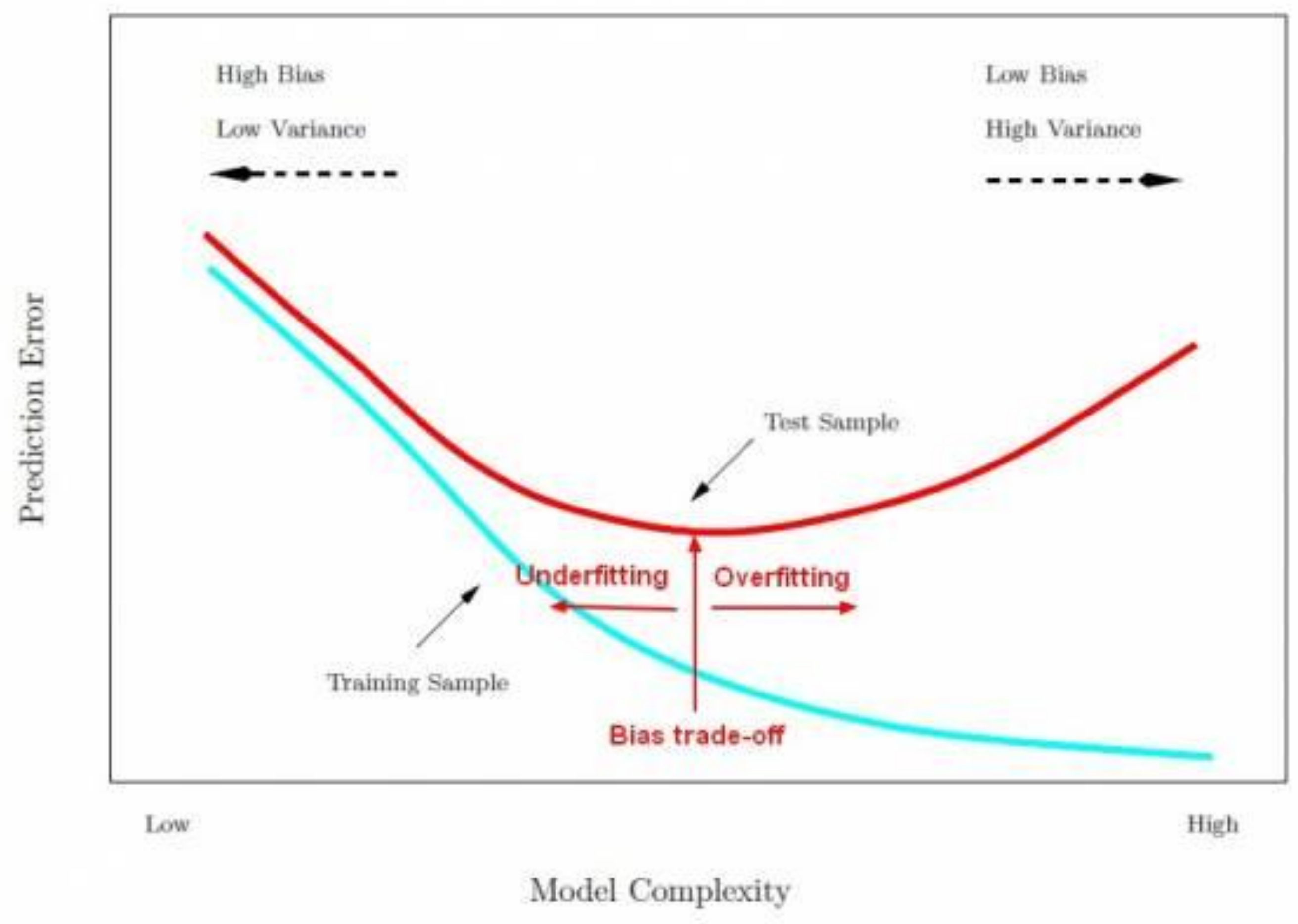
- Source: http://www.resacorp.com/lsmse2.htm

		Condition (as determined by "Gold standard")		
		Condition positive	Condition negative	
Test outcome	Test outcome positive	**True positive**	**False positive** (Type I error)	Precision = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Test outcome positive}}$
	Test outcome negative	**False negative** (Type II error)	**True negative**	Negative predictive value = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Test outcome negative}}$
		Sensitivity = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	**Specificity** = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$	Accuracy = $\frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$

- Source: http://en.wikipedia.org/wiki/Sensitivity_and_specificity

Basic Notions of ML

- Overfitting
  - Low error on training data
  - High error on (unseen/new) test data
  - The model does not generalize well from the training data on unseen data
  - Possible causes:
    - Too few training examples
    - Model too complex (too many parameters)
- Underfitting
  - High error both on training and test data
  - Model is too simple to capture the information in the training data



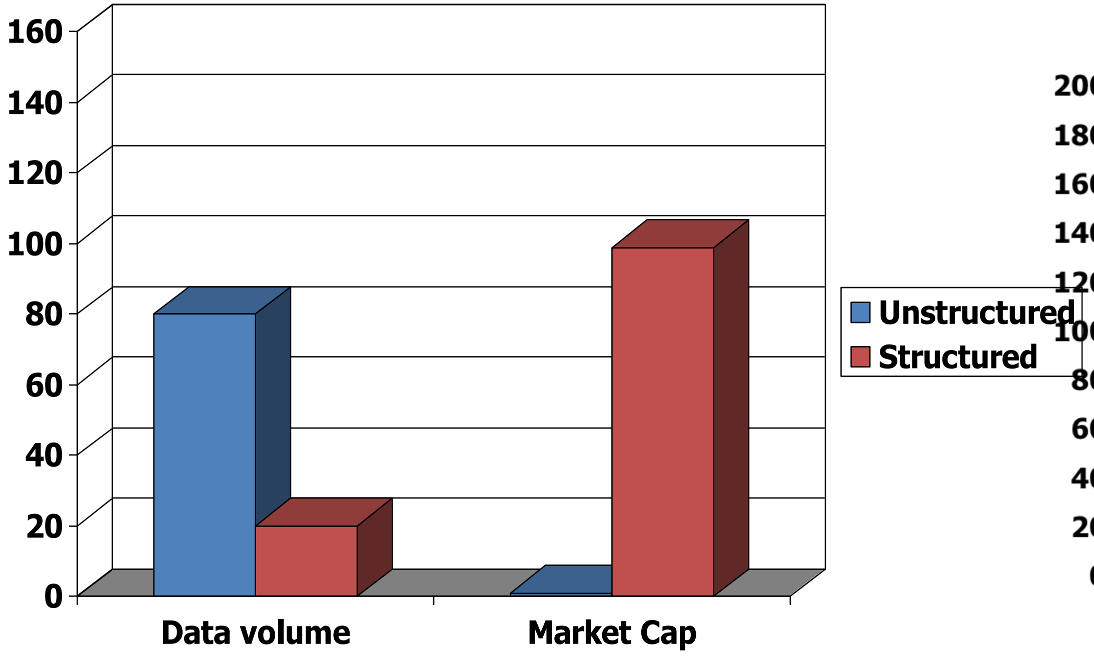
- Source: http://gerardnico.com/wiki/data_mining/overfitting

What is NLP?

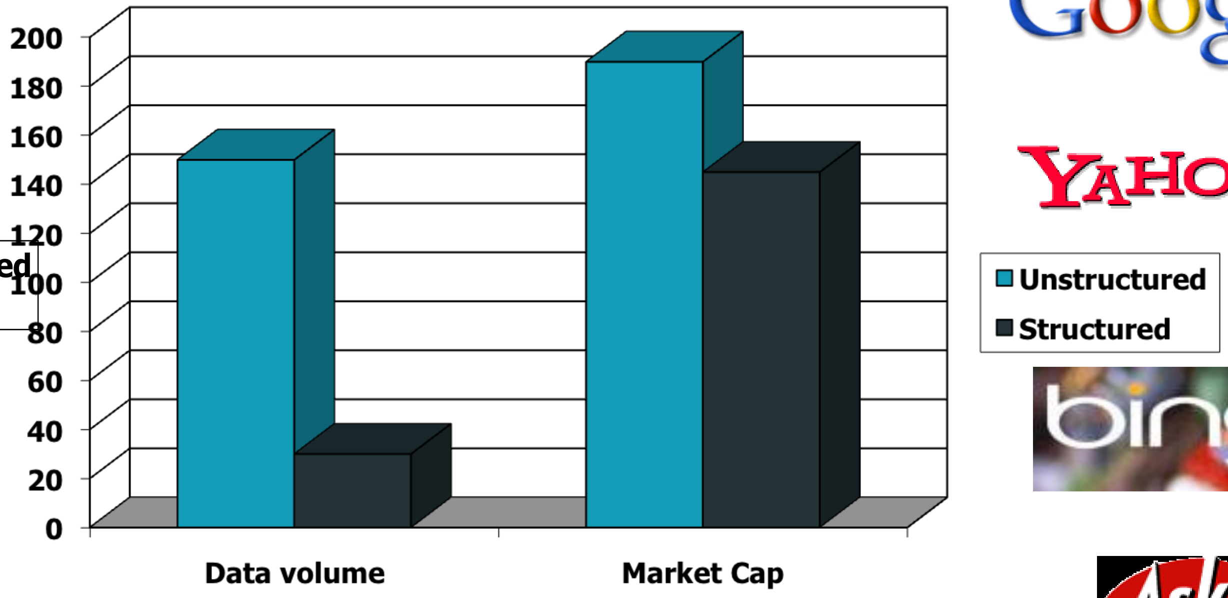
- Language = Words + Rules + Exceptions + More
  - dictionary (vocabulary) + grammar + more
- Dictionary
  - set of words defined in the language (static or dynamic)
- Grammar
  - set of rules which describe what is allowable in a language
- Natural Language
  - languages spoken by people (English, French, German, etc.) as opposed to artificial languages (C++, Java, Python, etc.) built for computer manipulation
- Natural Language Processing
  - computer applications that automatically analyze natural language

Why NLP?

- More and more unstructured data (e.g. text, images) available
- Understanding natural language is deeply linked with real AI (e.g. Turing test)



1996



2009

Why NLP ?

- 這是英文的一個非常簡單的文本。
- Αυτό είναι ένα πολύ απλό κείμενο στην αγγλική γλώσσα.
- Tämä on hyvin yksinkertainen teksti Englanti.
- هذا هو نص بسيط جدا في اللغة الإنجليزية.

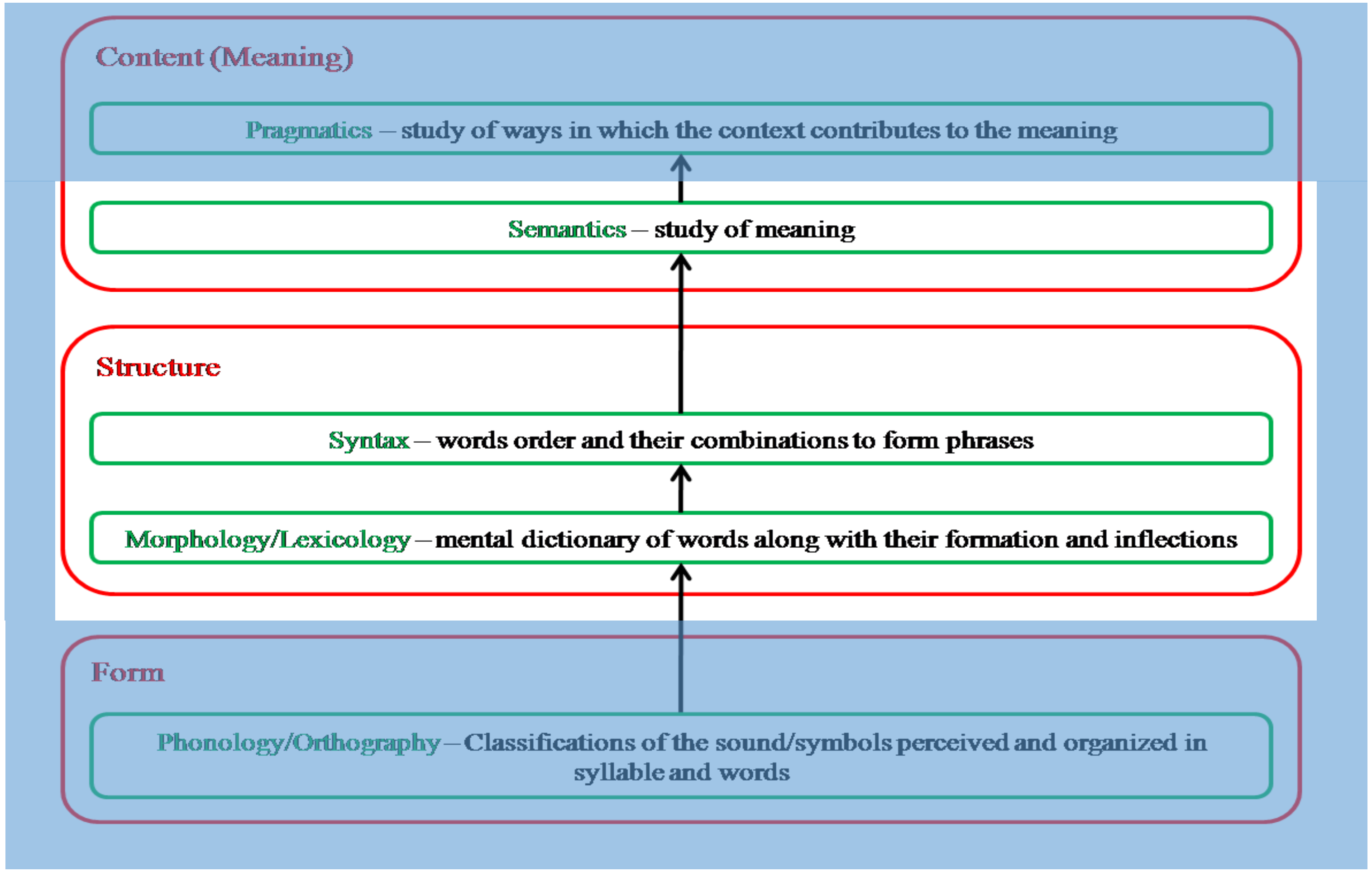
- Computers “see” text in natural languages the same you see the previous text!

- People have no trouble understanding language
  - Common sense knowledge
  - Reasoning capacity
  - Experience
- … but computers do!

Possible NLP applications

- Classify text documents into categories (e.g. customer e-mails, spam, etc.)
- Index and search large collections of texts (Information Retrieval, Google)
- Machine translation (Google Translate)
- Information extraction (Extract useful information from resumes)
- Automatic summarization (Condense a large text into a much smaller one without loosing relevant information)
- Question answering (Who was the 24th president of the USA?)
- Speech recognition (Understand phone conversations)
- Plagiarism detection (Detect if two text documents are very similar)
- Text proofreading – spelling & grammar (Spellcheckers)
- Conversational agents (Siri, Cortana, etc.)
- ….

Natural language is complex



Morphology

- Low-level NLP processing, receives as input a string of letters/symbols and outputs information about the words/tokens in the document

- Tokenization
  - process of breaking a stream of text up into tokens (= words, phrases, symbols, or other meaningful elements)
  - Typically performed at the “word” level
  - Not easy: *Hewlett-Packard*, *U.S.A.*, in some languages there is no “space” between words!

A sentence in Chinese	我喜欢新西兰花
Interpretation 1	我 喜欢 新西兰 花
Interpretation 2	我 喜欢 新 西兰花

Morphology

- Stemming
  - Reduces similar words to a given “stem”
  - E.g. *detects*, *detected*, *detecting*, *detect* => *detect* (stem).
  - Usually set of rules for suffix stripping
  - Most popular for English: Porter's Algorithm
  - 36% reduction in indexing vocabulary (English)
  - Linguistic correctness of resulting stems not necessary (sensitivities → *sensit)*
- Lemmatization
  - Uses a vocabulary and full morphological analysis of words
  - Aims to remove inflectional endings only
  - Return the base or dictionary form of a word, which is known as the *lemma*.
  - E.g. *saw => see,* *been, was => be*
- *Other language specific issues*
  - Split compound words (e.g. German)

Morphology

- POS tagging
  - POS = Part Of Speech
  - Determine for each word its grammatical category (whether it is a noun, adjective, verb, preposition, article, etc.)
  - POS tags = Represents a pretty much stable set across languages → Most commonly used POS sets for English have 50-80 different tags (Brown Corpus Tags)
- Very high accuracy (98%+ for English)
- Most words have only one POS tag!

Syntax

- Syntax captures structural relationships between words and phrases, i.e. describes the constituent structure of NL expressions
  - Constituents: Noun Phrase, Verb Phrase, Determiners....
- Grammars are used to describe the syntax of a language (Just like the syntax in programming languages) – e.g. next slide
  - structures and patterns in phrases
  - how phrases are formed by smaller phrases and words
- Syntactic analyzers assign a syntactic structure (parse tree) to a string on the basis of a grammar → syntactic analyzers are also called parsers
- Why parsing?
  - Identifying the structure is the first step towards understanding the meaning of the sentence or to comparing strings (for machine translation)

Syntax

- Sample English grammar

$S \rightarrow NP\ VP$	$Det \rightarrow that \mid this \mid a$
$S \rightarrow Aux\ NP\ VP$	$Noun \rightarrow book \mid flight \mid meal \mid money$
$S \rightarrow VP$	$Verb \rightarrow book \mid include \mid prefer$
$NP \rightarrow Pronoun$	$Pronoun \rightarrow I \mid she \mid me$
$NP \rightarrow Proper\text{-}Noun$	$Proper\text{-}Noun \rightarrow Houston \mid TWA$
$NP \rightarrow Det\ Nominal$	$Aux \rightarrow does$
$Nominal \rightarrow Noun$	$Preposition \rightarrow from \mid to \mid on \mid near \mid through$
$Nominal \rightarrow Nominal\ Noun$	
$Nominal \rightarrow Nominal\ PP$	
$VP \rightarrow Verb$	
$VP \rightarrow Verb\ NP$	
$VP \rightarrow Verb\ NP\ PP$	
$VP \rightarrow Verb\ PP$	
$VP \rightarrow VP\ PP$	
$PP \rightarrow Preposition\ NP$	

Parsing

- Sentence $\Rightarrow$ parse tree

- Input:
  - sequence of pairs (lemma, (morphological) tag)
- Output:
  - sentence structure (tree) with annotated nodes (all lemmas, (morphosyntactic) tags, functions), of various forms
- Deals with:
  - the relation between lemmas & morphological categories and the sentence structure
  - uses syntactic categories such as Subject, Verb, Object,…

How is parsing done?

- Augment the grammar with probabilities computed on corpora manually annotated by linguists
- Use dynamic programming to construct structures from substructures

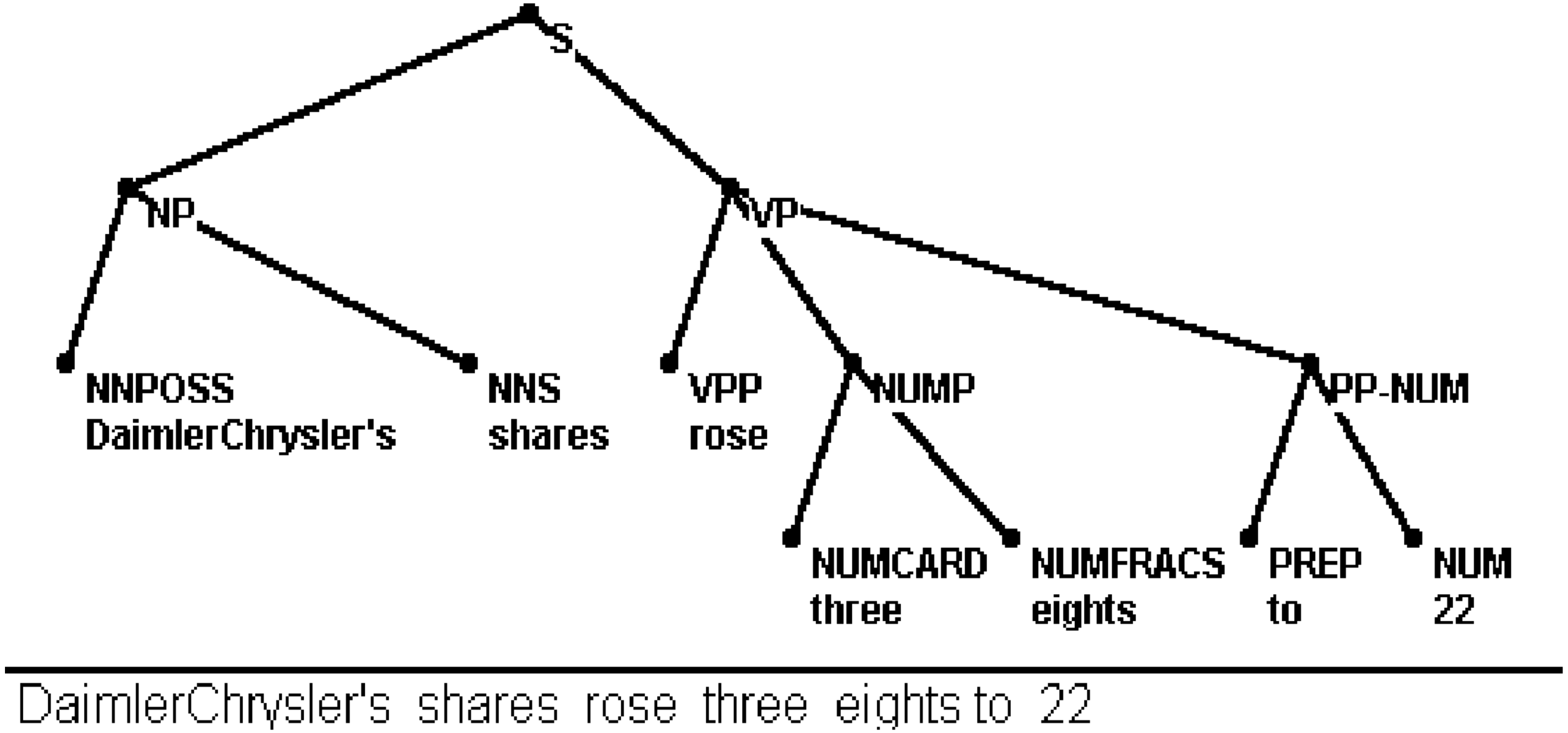
S → *NP VP*	[.80]	*Det* → *that* [.05] | *the* [.80] | *a* [.15]	
S → *Aux NP VP*	[.15]	*Noun* → *book*	[.10]
S → *VP*	[.05]	*Noun* → *flights*	[.50]
NP → *Det Nom*	[.20]	*Noun* → *meal*	[.40]
NP → *Proper-Noun*	[.35]	*Verb* → *book*	[.30]
NP → *Nom*	[.05]	*Verb* → *include*	[.30]
NP → *Pronoun*	[.40]	*Verb* → *want*	[.40]
Nom → *Noun*	[.75]	*Aux* → *can*	[.40]
Nom → *Noun Nom*	[.20]	*Aux* → *does*	[.30]
Nom → *Proper-Noun Nom*	[.05]	*Aux* → *do*	[.30]
VP → *Verb*	[.55]	*Proper-Noun* → *TWA*	[.40]
VP → *Verb NP*	[.40]	*Proper-Noun* → *Denver*	[.40]
VP → *Verb NP NP*	[.05]	*Pronoun* → *you* [.40] | *I* [.60]	

Syntax: Representation of the results

- Tree structure (“tree” in the sense of graph theory)
  - one tree per sentence

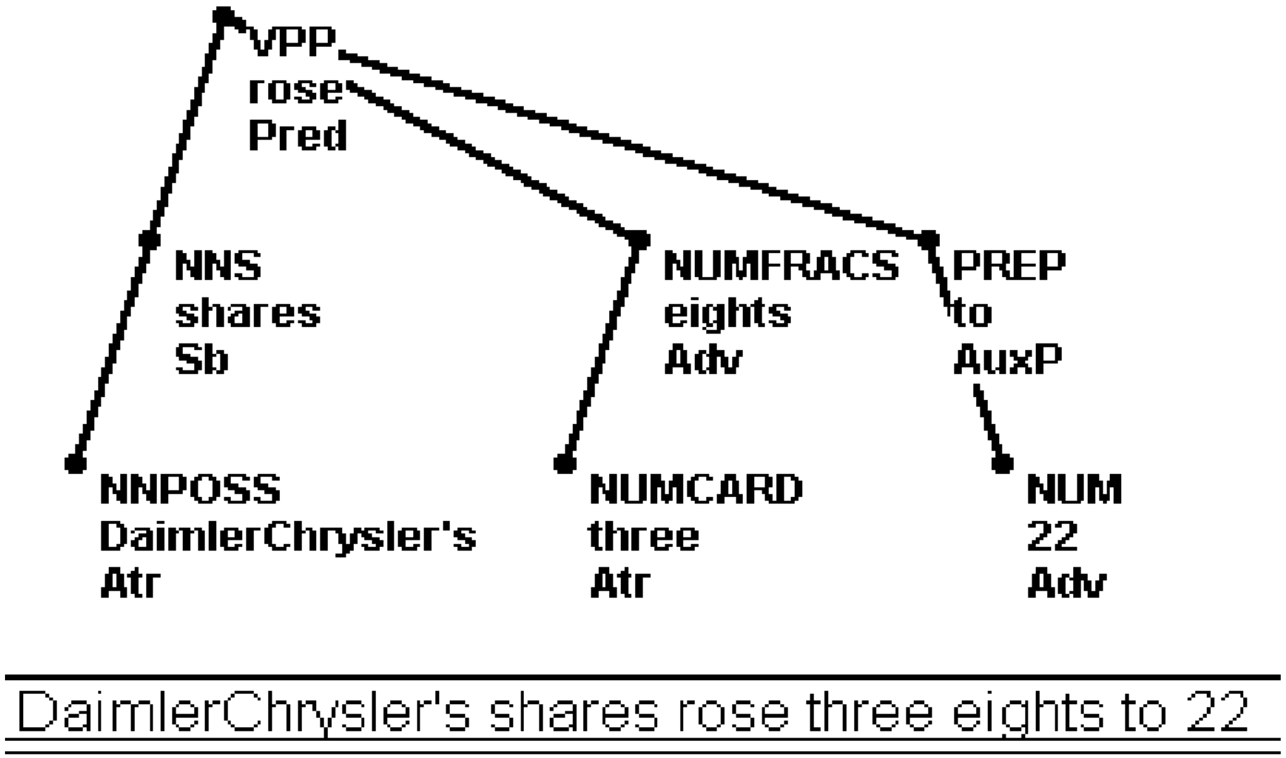
- Two main ideas for the shape of the tree:
  - phrase structure (~ derivation tree, cf. parsing later)
    - using bracketed grouping
    - brackets annotated by phrase type
    - heads (often) explicitly marked
  - dependency structure (lexical relations “local”, functions)
    - basic relation: head (governor) - dependent
    - links (edges) annotated by syntactic function (Sb, Obj, ...)
    - phrase structure: implicitly present

E.g.: Phrase Structure Tree



- ((DaimlerChrysler's shares)$_{NP}$ (rose (three eights)$_{NUMP}$ (to 22)$_{PP\text{-}NUM}$)$_{VP}$)$_{S}$

E.g.: Dependency Tree



- $rose_{Pred}(shares_{Sb}(DaimlerChrysler's_{Atr}),eights_{Adv}(three_{Atr}),to_{AuxP}(22_{Adv}))$

Semantics

- Semantics and pragmatics contain complex high-level NLP tasks
- Semantics = understanding "meaning" of words
- Pragmatics = language use in context (jokes, irony, dialogue related aspects, etc.)

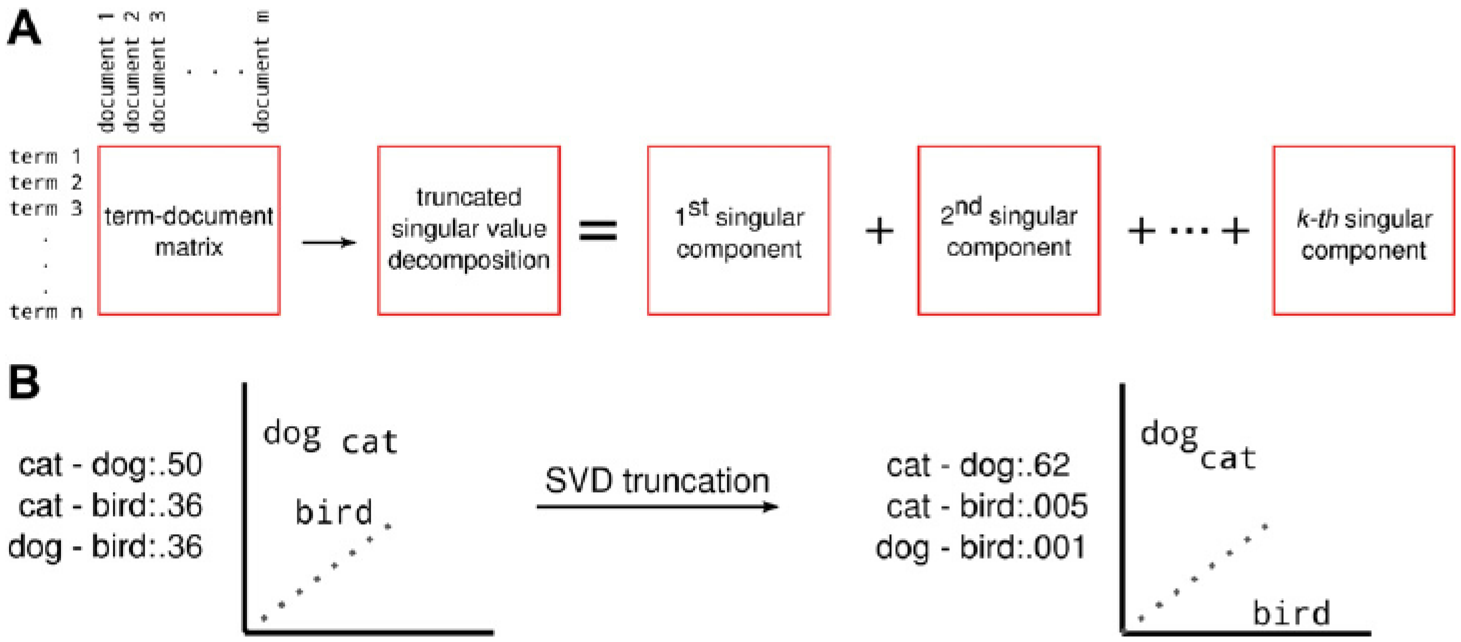
- What is the meaning of words?
  - Dictionary definitions?
  - Synonyms, antonyms, etc.
  - Is "car" related to "engine"? How about "car" and "gas"?
  - The meaning requires lots of common-sense knowledge – human specific

Semantics

- How can computers use/understand human knowledge?
- Either use human-made knowledge bases (called ontologies)
  - WordNet
  - FrameNet
  - Etc.
- Try to build knowledge bases on their own by analyzing large collections of texts (maybe use some human “seeds” for relations and concepts)
  - NELL (Never Ending Language Learning)
  - Probase
  - Freebase, Dbpedia
  - Google Knowledge Graph
- Try to assess meaning from the context of words
  - The meaning of a word is actually distributed in the “meaning” of the words used together with it
  - Compute these distributed word embeddings

Word embeddings

- Compute a vector representing the distributed representation for every word
- Various methods to do this
  - For example, Latent Semantic Analysis (LSA) uses Singular Value Decomposition (SVD)



Word2Vec

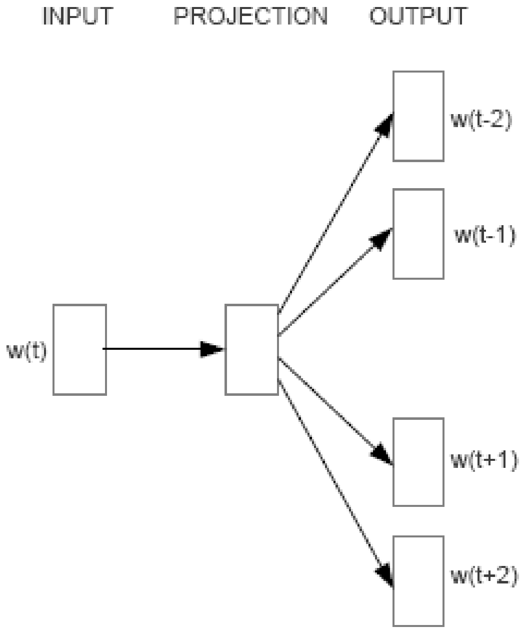
- Neural Network language model
- Released in 2013 by Google
- Advantages
  - Can compute word embeddings on datasets larger than any previous method
  - They seem to capture "subtle semantic relationships between words" (in the embedding space)

Type of relationship	Word Pair 1		Word Pair 2	
Common capital city	Athens	Greece	Oslo	Norway
All capital cities	Astana	Kazakhstan	Harare	Zimbabwe
Currency	Angola	kwanza	Iran	rial
City-in-state	Chicago	Illinois	Stockton	California
Man-Woman	brother	sister	grandson	granddaughter
Adjective to adverb	apparent	apparently	rapid	rapidly
Opposite	possibly	impossibly	ethical	unethical
Comparative	great	greater	tough	tougher
Superlative	easy	easiest	lucky	luckiest
Present Participle	think	thinking	read	reading
Nationality adjective	Switzerland	Swiss	Cambodia	Cambodian
Past tense	walking	walked	swimming	swam
Plural nouns	mouse	mice	dollar	dollars
Plural verbs	work	works	speak	speaks

Continuous Skip-gram Model

- Instead of predicting the current word based on the context
- Tries to maximize classification of a word based on another word in the same sentence
- Thus, uses each current word as an input to a log-linear classifier
- Predicts words within a certain window
- Observations
  - Larger window size => better quality of the resulting word vectors, higher training time
  - More distant words are usually less related to the current word than those close to it
  - Give less weight to the distant words by sampling less from those words in the training examples

Continuous Skip-gram Model

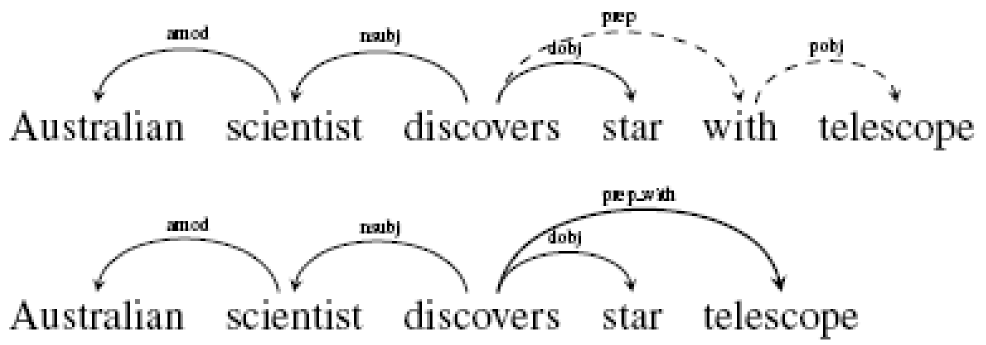


Skip-gram

$$Q = C \times (D + D \times log_2(V)),$$

Dependency-based Contexts

- Levi and Goldberg, 2014: Propose to use dependency-based contexts instead of linear BoW (windows of size k)



WORD	CONTEXTS
australian	scientist/amod^{-1}
scientist	australian/amod, discovers/nsubj^{-1}
discovers	scientist/nsubj, star/dobj, telescope/prep_with
star	discovers/dobj^{-1}
telescope	discovers/prep_with^{-1}

Finding vandalism on Wikipedia

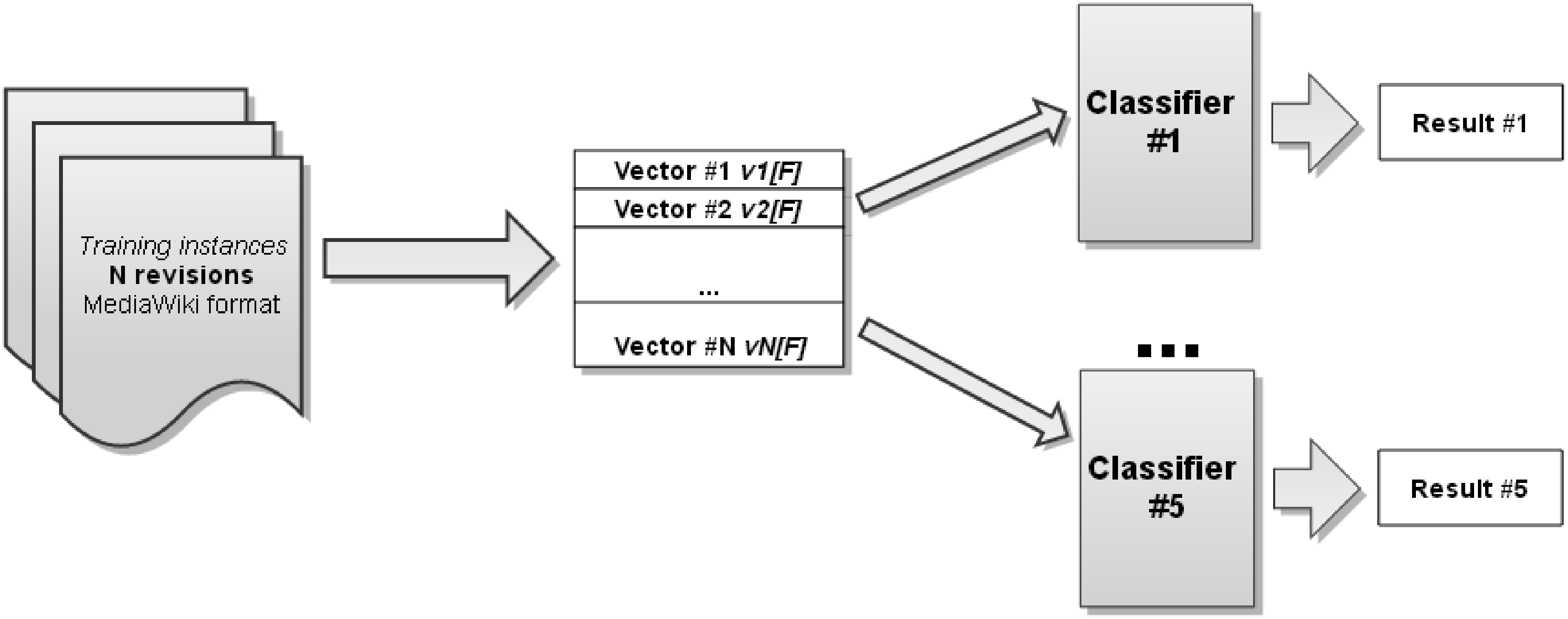
- Work with Dan Cioiu
- **Vandalism** = editing in a malicious manner that is intentionally disruptive
- Dataset used at PAN 2011
  - Real articles extracted from Wikipedia in 2009 and 2010
  - Training data: 32,000 edits
  - Test data: 130,000 edits
- An edit consists of:
  - the old and new content of the article
  - author information
  - revision comment and timestamp
- We chose to use only the actual text in the revision (no author and timestamp information) to predict vandalism

Features

variation of pronouns in the article body	ratio between current and past revision lengths
length of longest sequence of identical consecutive characters	**length of longest word from the article**
number of added characters	**number of suspicious (but non-vulgar) words added to the article**
whether or not the edit is a single-word update	number of new blocks of HTML comments
number of consecutive character sequences added	**whether the user is anonymous**
number of vulgar words added to the text	ratio between vulgar words and total words in the edit comments
number of added pronouns	variation of vulgar words
ratio between number of added uppercase letters and total added letters	number of identical character sequences added

similarity measure between added text and initial text	similarity measure between added deleted text and resulting text
semantic similarity measure between added text and initial text	semantic similarity measure between deleted text and resulted text

Using several classifiers



Cross-validation results

BayesianLogisticRegression (84.5028 % correctly identified)

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.935	0.438	0.87	0.935	0.902	0.749	0.0
	0.562	0.065	0.735	0.562	0.637	0.749	1.0
Weighted Avg.	0.845	0.348	0.837	0.845	0.838	0.749	

BayesNet (85.3508 % correctly identified)

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.879	0.227	0.924	0.879	0.901	0.908	0.0
	0.773	0.121	0.671	0.773	0.718	0.908	1.0
Weighted Avg.	0.854	0.201	0.863	0.854	0.857	0.908	

NBTree (86.9965 % correctly identified)

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.92	0.288	0.909	0.92	0.915	0.907	0.0
	0.712	0.08	0.74	0.712	0.726	0.907	1.0
Weighted Avg.	0.87	0.238	0.868	0.87	0.869	0.907	

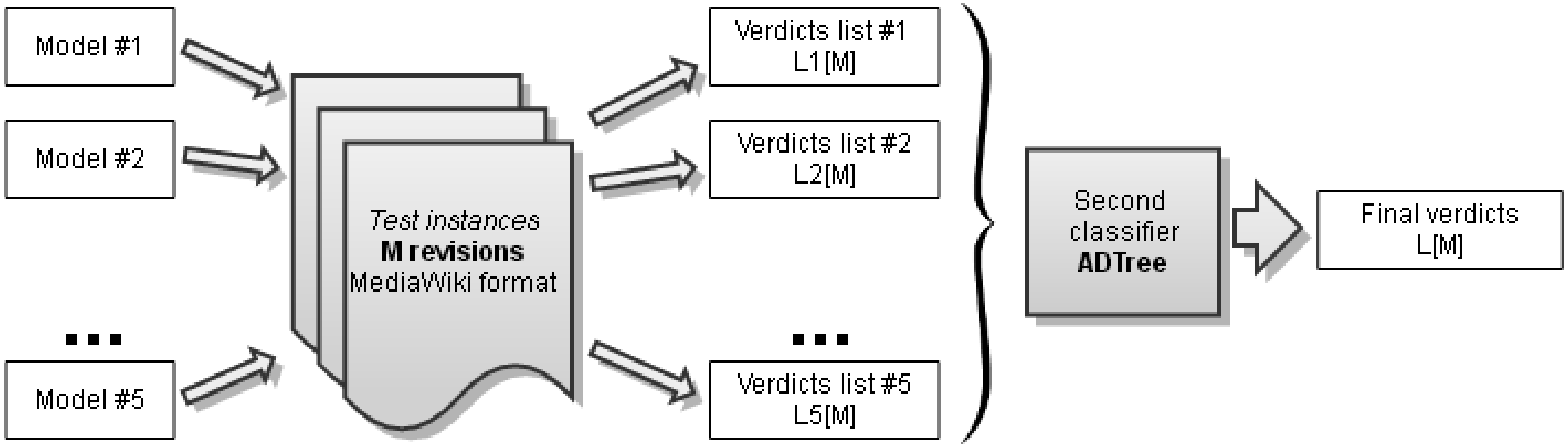
SVM (84.2908 % correctly identified)

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.932	0.436	0.87	0.932	0.9	0.748	0.0
	0.564	0.068	0.725	0.564	0.634	0.748	1.0
Weighted Avg.	0.843	0.347	0.835	0.843	0.836	0.748	

ADTree (86.4917 % correctly identified)

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.927	0.331	0.898	0.927	0.912	0.911	0.0
	0.669	0.073	0.746	0.669	0.705	0.911	1.0
Weighted Avg.	0.865	0.268	0.861	0.865	0.862	0.911	

Combining results using a meta-classifier



Final results on test set

ROC-AUC	PR-AUC	Detector
0.92236	0.66522	Mola Velasco, 2010
0.90351	0.49263	Adler et al., 2010
0.89856	0.44756	Javanmardi, 2010
0.89377	0.56213	Chichkov, 2010
0.89375	**0.48294**	**WikiDetect**
0.87990	0.41365	Seaward, 2010
0.87669	0.42203	Hagedus et al., 2010
0.85875	0.41498	Harpalani et al., 2010
0.84340	0.39341	White et al., 2010
0.65404	0.12235	Iftene, 2010

- Source: http://link.springer.com/chapter/10.1007%2F978-3-642-40495-5_32

Main conclusions

- **Feature extraction is important in ML**
  - Which are the best features?
- Syntactic and context analysis are the most important
- Semantic analysis is slow but increases detection by 10%
- Using features related to users would further improve our results
- **A meta-classifier sometimes improves the results of several individual classifiers**
  - Not always! Maybe another discussion on this…

Sexual predator detection

- Work with Claudia Cardei
- Automatically detect sexual predators in a chat discussion in order to alert the parents or the police
- Corpus from PAN 2012
  - 60000+ chat conversations (around 10000 after removing outliers)
  - 90000+ users
  - 142 sexual predators
- Sexual predator is used “to describe a person seen as obtaining or trying to obtain sexual contact with another person in a metaphorically *predatory* manner” (Wikipedia)

Problem description

- Decide either a user is a predator or not
- Use only the chats that have a predator (balanced corpus: 50% predators - 50% victims)
- One document for each user
- How to determine the features?

Simple solution

- "Bag of words" (BoW) features with tf-idf weighting
- Feature extraction using mutual information

- MLP (neural network) with 500 features: 89%

Going beyond

- The should be other features that are discriminative for predators
- Investigated behavioral features:
  - Percentage of questions and inquiries initiated by an user
  - Percentage of negations
  - Expressions that might denote an underage user (“*don’t know*”, “*never did*”,…)
  - The age of the user if found in a chat reply (“*asl*” like answers)
  - Percentage of slang words
  - Percentage of words with a sexual connotation
  - Readability scores (Flesch)
  - Who started the conversation

- AdaBoost (with DecisionStump): 90%
- Random Forest: 93%

Main conclusions

- **Feature selection is useful when the number of features is large**
  - Reduced from 10k+ features to hundreds/thousands
  - We used MI, there are others techniques
- **Fewer features can be as descriptive** as (or even **more descriptive** than) the BoW model
  - Difficult to choose
  - Need additional computations and information
  - Need to understand the dataset

Practical ML

- Brief discussion of alternatives that might be used by a programmer
- The main idea is to show that it is simple to start working on a ML task
- However, it is more difficult to achieve very good results
    - Experience
    - Theoretical underpinnings of models
    - Statistical background

R language

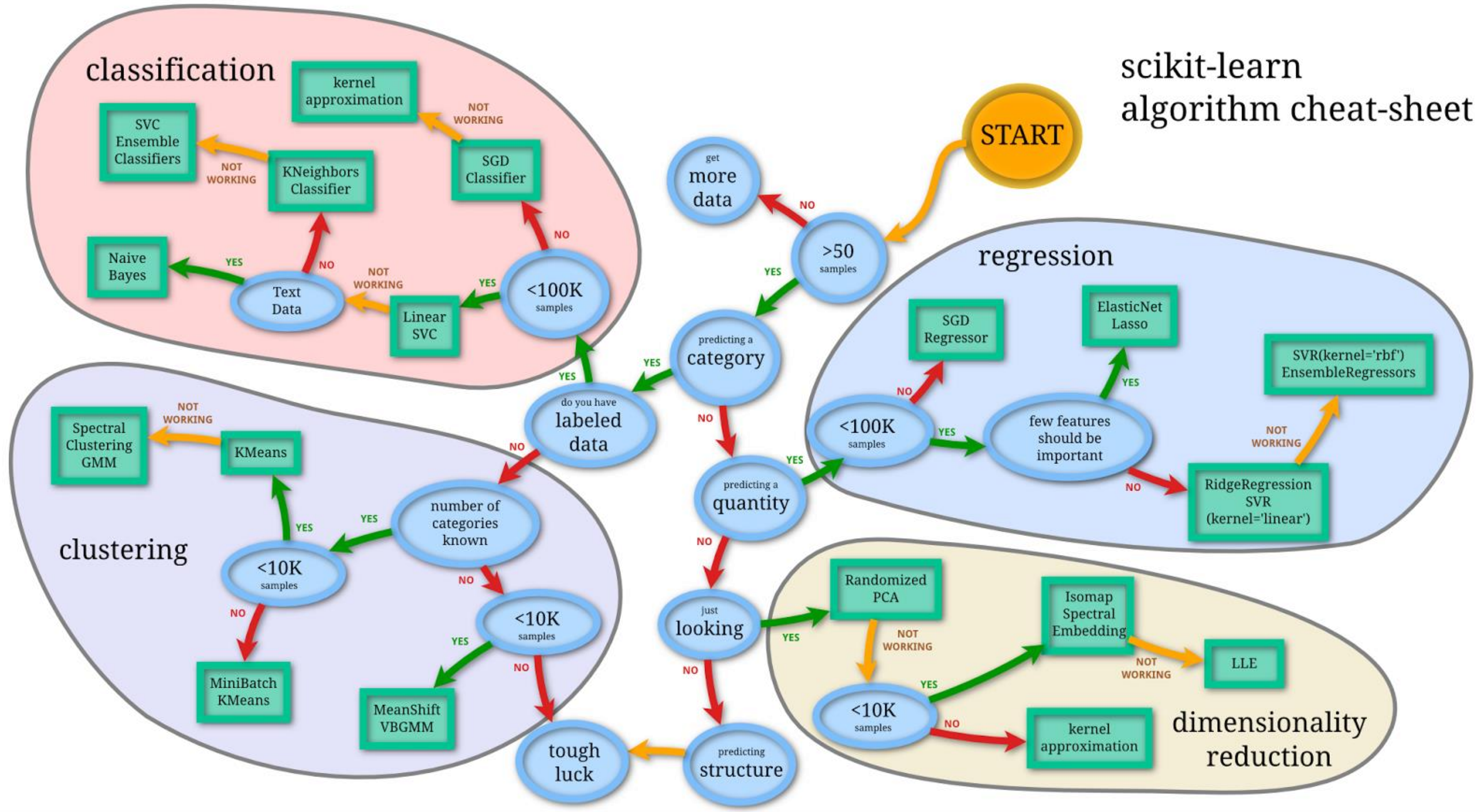
- Designed for “statistical computing”

- Lots of packages and functions for ML & statistics
- Simple to output visualizations
- Easy to write code, short

- It is kind of slow on some tasks
- Need to understand some new concepts: data frame, factor, etc.
- You may find the same functionality in several packages (e.g. NaïveBayes, Cohen’s Kappa, etc.)

Python (sklearn)

- sklearn (scikit-learn) package developed for ML
- Uses existing packages numpy, scipy
- Started as a GSOC project in 2007
- It is new, implemented efficiently (time and memory), lots of functionality
- Easy to write code
  - Naming conventions, parameters, etc.



- Source: http://peekaboo-vision.blogspot.ro/2013/01/machine-learning-cheat-sheet-for-scikit.html

Java (Weka)

- Similar to scikit-learn, but developed in Java
- Older (since 1997) and more stable than scikit-learn
- Some implementations are (were) a little bit more inefficient, especially related to memory consumption
- However, usually it is fast, accurate, and has lots of other useful utilities
- It also has a GUI for getting some results fast without writing any code

Not discussed

- RapidMiner
  - Easy to use
  - GUI interface for building the processing pipeline
  - Can write some code/plugins
- Matlab/Octave
  - Numeric computing
  - Paid vs. free
- SPSS/SAS/Stata
  - Somehow similar to R, faster, more robust, expensive
- Many others:
  - Apache Mahout / MALLET / NLTK / Orange / MLPACK
  - Specific solutions for various ML models

Practical NLP

- Again, there are a wide variety of open-source tools for NLP
- In most programming languages
- We present only a few, most popular
- They contain set of tools which form a NLP pipeline (at least morphology & syntax, some semantics and pragmatics)
- Stanford CoreNLP
- Python NLTK
- Cython spaCy – newer
- There are several other specific tools (e.g. for Word2Vec, LSA, WordNet, etc.) and NLP packages (OpenNLP, Lingpipe)

Stanford CoreNLP

- Implemented in Java
- Widely used in research and commercial products
- Pipeline: sentence splitter, tokenizer, lemmatizer, POS tagger, (syntactic and dependency) parser, corefernce resolution, named-entity recognition, etc.
- Available for several languages (English, Chinese, Spanish, French, German)
- Link: http://stanfordnlp.github.io/CoreNLP/
- Demo: http://nlp.stanford.edu:8080/corenlp/

NLTK

- Natural Language ToolKit
- Available in Python
- Lots of implemented modules (full list here: http://www.nltk.org/py-modindex.html)
- Provides the same functionality as CoreNLP and even more (can use CoreNLP if needed)
- Also implements several classifier and some simple conversational agents

SpaCy

- Newer, faster (at least this in their benchmarks) than previous solutions
- Implemented in Cython
- “high performance tokenizer, part-of-speech tagger, named entity recognizer and syntactic dependency parser, with built-in support for word vectors”
- Link: https://spacy.io/

Other

- Gensim (https://radimrehurek.com/gensim/): for computing word embeddings, including Word2Vec
- SyntaxNet (https://research.googleblog.com/2016/05/announcing-syntaxnet-worlds-most.html) : syntactic parser from Google, released in 2016 , very accurate
- Lots more ☺

Conclusions

- Lots of interesting ML+NLP tasks and data available
  - Do you have access to new and interesting data in your business?
- Several programming alternatives, easy to use

- Understand the basics
- Enhance your Maths (statistics) skills for a better ML experience
- Hands-on experience and communities of practice
  - www.kaggle.com (and other similar initiatives)

Thank you!

traian.rebedea@cs.pub.ro

References and Resources - NLP

- Rada Mihalcea. 2011. Natural Language Processing. CSCE 5290 Natural Language Processing, Course overview, 2011 (http://www.cse.unt.edu/~rada/CSCE5290/Lectures/Intro.ppt)
- Rada Mihalcea. Linguistics Essentials. CSCE 5290 Natural Language Processing, Course overview, 2011 (www.cse.unt.edu/~rada/CSCE5290/Lectures/Ling.Essentials.ppt)
- Fabienne Venant. Chunking / partial parsing. Université Nancy2 / Loria
- Stefan Trausan-Matu. Introduction to Morphology.
- James Pustejovsky. 2008. Grammar and Parsing (II). CS 114 Introduction to Computational Linguistics
- Igo Dagan. Part of Speech tagging. Statistical and Learning Methods in Natural Language Processing, Spring 2004 (http://cs.haifa.ac.il/~shuly/teaching/04/statnlp/pos-tagging.ppt)
- Igo Dagan & Shuly Wintner. Shallow parsing. Statistical and Learning Methods in Natural Language Processing, Spring 2004 (http://cs.haifa.ac.il/~shuly/teaching/04/statnlp/chunking.pdf)
- Zeynep ORHAN. Opportunities in Natural Language Processing. CENG 502: Special Topics in Computer Engineering (www.fatih.edu.tr/~zorhan/ceng502/lecturenotes/NLPmyintro.ppt)

References and Resources – Word2Vec

- Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.
- Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems* (pp. 3111-3119).
- Mikolov, T., Yih, W. T., & Zweig, G. (2013, June). Linguistic Regularities in Continuous Space Word Representations. In *HLT-NAACL* (pp. 746-751).

Applications of word2vec

- Mikolov, T., Le, Q. V., & Sutskever, I. (2013). Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*.
- Levy, O., & Goldberg, Y. (2014). Dependency based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics* (Vol. 2, pp. 302-308).