

Introduction to NLP in an Interdisciplinary Setting

Lect. Dr. Ana-Sabina Uban

auban@fmi.unibuc.ro

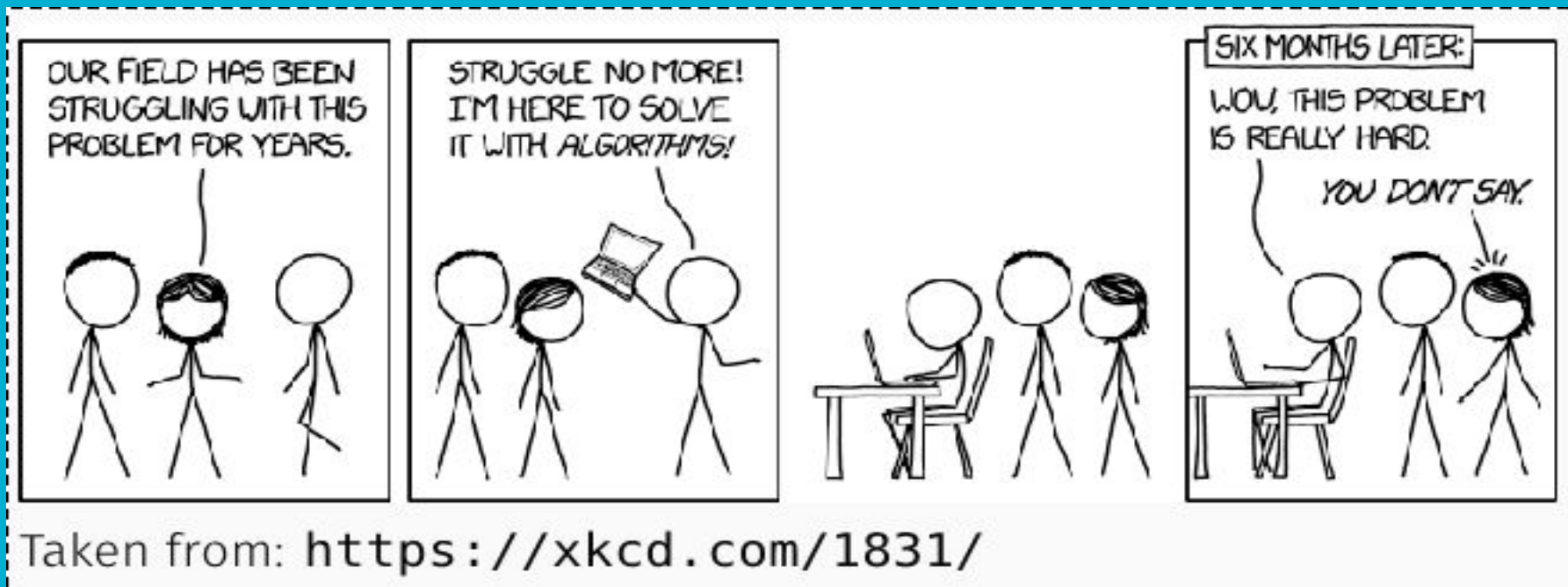
Main References

1. Dan Jurafsky and James H. Martin. *Speech and Language Processing* (3rd ed. draft)
(<https://web.stanford.edu/~jurafsky/slp3/>)
2. Chris Manning and Hinrich Schütze, *Foundations of Statistical Natural Language Processing*, MIT Press. Cambridge, MA: May 1999.
(<https://nlp.stanford.edu/fsnlp/>,
https://www.cs.vassar.edu/~cs366/docs/Manning_Schuetze_StatisticalNLP.pdf)
3. <https://www.aclweb.org/anthology/>

Interdisciplinarity

- Not aiming to beat a chess champion at tennis or viceversa!
- (predominant) Quantitative + (predominant) Qualitative
 - Examples
 - Biology + Informatics
 - Psychology + linguistics + informatics

Interdisciplinarity



Additional bibliography

● Stanford:

- <https://nlp.stanford.edu/IR-book/html/htmledition/tokenization-1.html>
- <https://nlp.stanford.edu/IR-book/html/htmledition/contents-1.html>

● Blogs:

- <http://mlexplained.com/2019/11/06/a-deep-dive-into-the-wonderful-world-of-preprocessing-in-nlp/>
(recommended)
- <https://medium.com/data-science-in-your-pocket/tokenization-algorithms-in-natural-language-processing-nlp-1fceb8454af>
- <https://towardsdatascience.com/tokenization-for-natural-language-processing-a179a891bad4>
- <https://blog.floydhub.com/tokenization-nlp/>
- https://dair.ai/notebooks/nlp/2020/03/19/nlp_basics_tokenization_segmentation.html

Additional bibliography

For some blueprints with introductory examples of text preprocessing and building machine learning models trained on textual data, you can see:

<https://github.com/ananana/nlp-basics>

For details on more advanced methods and instructions on reading/reviewing scientific research & project report structure:

<https://github.com/ananana/nlp-projects/blob/main/README.md>

Main steps

- Machine learning models need numerical inputs, so preprocessing is basically the process of taking a raw chunk of text and converting it into numbers.
- Therefore, for most applications, preprocessing can roughly be divided into the following 3 steps:
 - **Step 1: Normalization (Cleaning)**
 - **Step 2: Segmentation (Tokenization)**
 - **Step 3: Numericalization (Vocabulary mapping)**

Definitions

- **Normalization**: is where we clean the data in advance to remove unwanted inputs and to convert certain characters/sequences into canonical forms.
- **Tokenization** is breaking a text chunk in smaller parts. Whether it is breaking paragraphs into sentences, sentences into words or words into characters.
- **Numericalization**: is where we convert the textual entities into numbers/ids so that we can feed them to our model.

Errors

- In NLP we are always dealing with errors.
- Reducing the error rate for an application often involves two antagonistic efforts:
 - Increasing accuracy or precision (minimizing false positives)
 - Increasing coverage or recall (minimizing false negatives).

Step 1: Normalization (Cleaning)

- It rather refers to the process of cleaning up the input and mapping characters/words to a "canonical" form.
 - Example: mapping all characters to their lowercase form (Ciao=ciao).
- **Standard steps:**
 - Handling repeating characters (e.g. "goooooood" -> "good")
 - Handling homoglyphs (e.g. "\$tupid" -> "stupid")
 - Mapping special inputs such as URLs, email addresses, and HTML tags to a canonical form (e.g. "http://www.foo.com/bar" -> "[URL]")
 - Unicode normalization

Processing text data

Preprocessing:
Normalization, Tokenization
(Segmentation),
lemmatization, etc

Why?

- Preprocessing: one of the most undervalued and overlooked elements of NLP
- Although there are blog posts abound on state of the art models and optimizers, there aren't many posts on how to tokenize the input.
- This is probably partly because it seems so easy, and partly because it just isn't sexy.
- The apathy towards preprocessing is even more evident in academia

Main difficulty:

Language dependent!

Domain dependent!

Normalization

- Need to “normalize” terms
 - Information Retrieval: indexed text & query terms must have same form.
 - We want to match *U.S.A.* and *USA*
- We implicitly define equivalence classes of terms
 - e.g., deleting periods in a term
- Alternative: asymmetric expansion:
 - Enter: *window* Search: *window, windows*
 - Enter: *windows* Search: *Windows, windows, window*
 - Enter: *Windows* Search: *Windows*
- Potentially more powerful, but less efficient

Case folding

- Applications like IR: reduce all letters to lower case
 - Since users tend to use lower case
 - Possible exception: upper case in mid-sentence?
 - e.g., *General Motors*
 - *MeSH* vs. *mesh*
 - *US* vs. *us*
- For sentiment analysis, MT, Information extraction
 - Case is helpful (*US* versus *us* is important)

Lemmatization, morphology?

- Reduce inflections or variant forms to base form
 - *am, are, is* → *be*
 - *car, cars, car's, cars'* → *car*
- *the boy's cars are different colors* → *the boy car be different color*
- Lemmatization: have to find correct dictionary headword form
- Machine translation
 - Spanish *quiero* ('I want'), *quieres* ('you want') same lemma as *querer* 'want'

Choosing Normalization Steps

- What kinds of normalization should we actually apply?
 - No clear answers to this, but:
- Always look at the data manually!
- Are you using a pretrained model (e.g. BERT)?
 - If so, make sure your preprocessing steps match the preprocessing that is conducted for pretraining. This can be more difficult than it seems, since papers don't always report all their preprocessing steps.

In the end, depends on the data, the application and the model used.

Normalization choices affect the end result.

• • •

—

- How much crucial information does normalization remove?
 - In social media, saying "HELLO" and "Hello" can have different nuances. At the same time, treating "Hello world" and "hello world" differently might not make much sense. Sometimes, capitalization can indicate important grammatical information, such as the fact that a word is a proper noun (e.g. "New York").
- How much data do you have?
 - If you have massive amounts of data, you will probably need less normalization since the model could just learn that "Hello" and "hello" are the same underlying word from their distributions. But for most NLP applications you don't have such a large corpus of data, so you might want to err on the side of more normalization.

• • •

—

- Is a large vocabulary size detrimental to your application?
 - Less normalization tends to lead to a larger vocabulary, though this does depend on what kind of tokenization you use. If you are training a generative model (classifying each output as one of V words in the vocabulary), a larger vocabulary size can slow down training significantly. It can also cause memory errors.

Segmentation / Tokenization

- Segmentation/Tokenization is probably the most complex part of the preprocessing pipeline.
- Naive tokenization algorithms,
- Rule-based tokenizers,
- modern (subword) tokenizers that are learned on data.

What is a word?

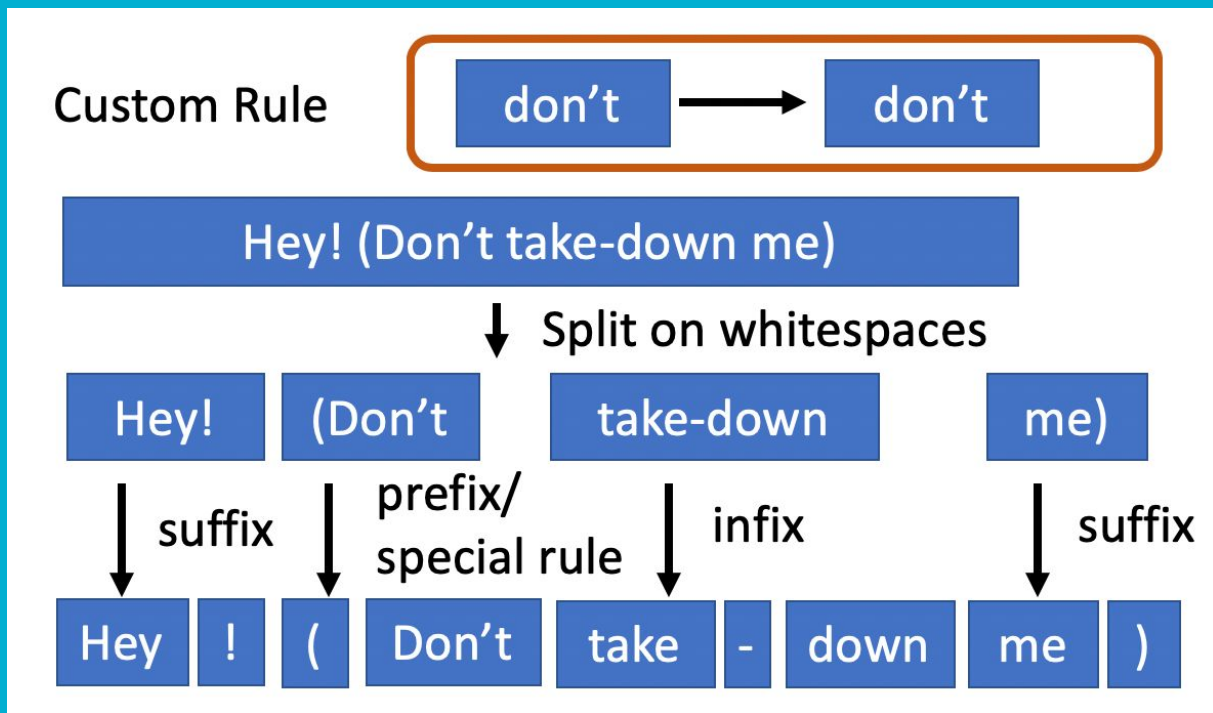
- What counts as a word is a vexed question in linguistics (there are words at various levels, such as phonological words versus syntactic words, which need not all be the same.)
- Kucera and Francis (1967) suggested the practical notion of a graphic word which they define as *string of contiguous alphanumeric characters with space on either side; may include hyphens and apostrophes, but no other punctuation marks.*
- What about \$22.50, 😊, ☹️, ...?

How many words?

- *I do uh main- mainly business data processing*
 - *Fragments, filled pauses*
- *Seuss's cat in the hat is different from other cats!*
 - **Lemma:** same stem, part of speech, rough word sense
 - cat and cats = same lemma
 - **Wordform:** the full inflected surface form
 - cat and cats = different wordforms

Rule-based Tokenization

- Allow us to tokenize more intelligently on a case-by-case basis



Sentence Segmentation

- What is a sentence?
- The first answer to what is a sentence is something **ending** with a “.” or “!”
- !, ? are relatively unambiguous
- Period “.” is quite ambiguous
 - Sentence boundary
 - Abbreviations like Inc. or Dr.
 - Numbers like .02% or 4.3
- Build a binary classifier
 - Looks at a “.”
 - Decides EndOfSentence/NotEndOfSentence
 - Classifiers: hand-written rules, regular expressions, or machine-learning

Lemmatizations. Definitions

- Variations of a word are called **wordforms** or **surface forms**.
- By reducing these wordforms to a common root, we simplify the input. The root form is called **lemma**.
- Lemmatization is the task of determining that two words have the same root, despite their surface differences.
 - *am, are, and is* have the shared lemma *be*;
 - *dinner* and *dinners* both have the lemma *dinner*.
 - He is reading detective stories -> *He be read detective story*.
- An algorithm or program that determines lemmas from wordforms is called a **lemmatizer**.

Examples

- Shakespeare's works: about 880K words, 29K wordforms, and 18K lemmas.
- Lemmatization involves word morphology, which is the study of word forms.
- Typically, we identify the morphological tags of a word before selecting the lemma.

Why?

- Many NLP tasks can benefit from lemmatization.
- Examples:
 - topic modelling looks at word distribution in a document.
 - By normalizing words to a common form, we get better results (In word embeddings removing inflected wordforms can improve downstream NLP tasks.)
 - For information retrieval (IR), lemmatization helps with query expansion so that suitable matches are returned even if there's not an exact word match.
 - In document clustering, it's useful to reduce the number of tokens. It also helps in machine translation.
- The decision to use lemmas is application dependent; we should use lemmas only if they show better performance.

Lemmatization vs stemming

Stemming

adjustable → adjust
formality → formaliti
formaliti → formal
airliner → airlin ⚠

Lemmatization

was → (to) be
better → good
meeting → meeting

Stemming

- Reduce terms to their stems in information retrieval
- *Stemming* is crude chopping of affixes
 - language dependent
 - e.g., *automate(s)*, *automatic*, *automation* all reduced to *automat*.

*for example compressed
and compression are both
accepted as equivalent to
compress.*



*for exampl compress and
compress ar both accept
as equival to compress*

Pre-processing for NLP

Normalization: is where we clean the data in advance to remove unwanted inputs and to convert certain characters/sequences into canonical forms.

Tokenization is breaking a text chunk in smaller parts. Whether it is breaking Paragraph in sentences, sentence into words or word in characters.

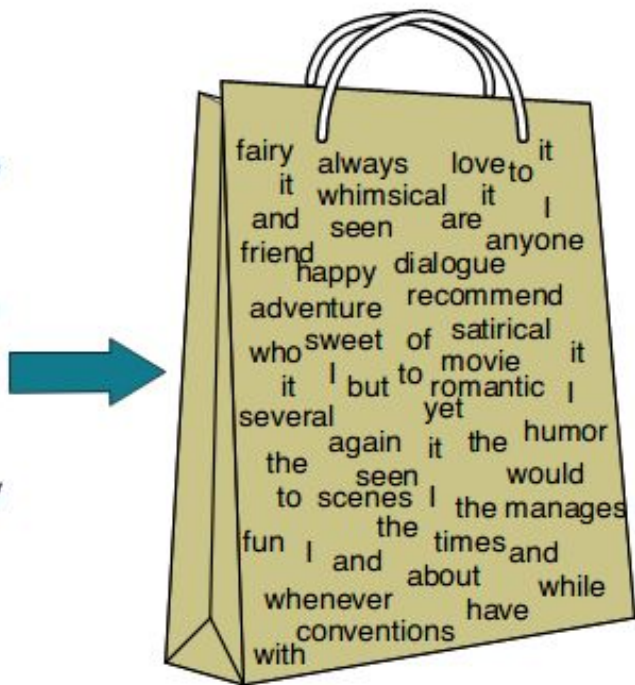
Numericalization: is where we convert the textual entities into numbers/ids so that we can feed them to our model.

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.



I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!



it	6
I	5
the	4
to	3
and	3
seen	2
yet	1
would	1
whimsical	1
times	1
sweet	1
satirical	1
adventure	1
genre	1
fairy	1
humor	1
have	1
great	1
...	...

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.

But raw frequency is a bad representation

Frequency is clearly useful; if *sugar* appears a lot near *apricot*, that's useful information.

But overly frequent words like *the*, *it*, or *they* are not very informative about the context

Need a function that resolves this frequency paradox!

Query: the movie that I liked the most

Document1: The cat is a domestic species of small carnivorous mammal. It is the only domesticated species in the family Felidae and is often referred to as the ...

Document 2: Stuart Little: best movie ever!

Term frequency tf

- The term frequency $tf_{t,d}$ 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

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*.

tf-idf: combine two factors

tf: term frequency. frequency count (usually log-transformed):

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

Idf: inverse document frequency: tf-

$$\text{idf}_i = \log \left(\frac{N}{\text{df}_i} \right)$$

Total # of docs in collection

of docs that have word i

Words like "the" or "good" have very low idf

tf-idf value for word t in document d:

$$w_{t,d} = \text{tf}_{t,d} \times \text{idf}_t$$

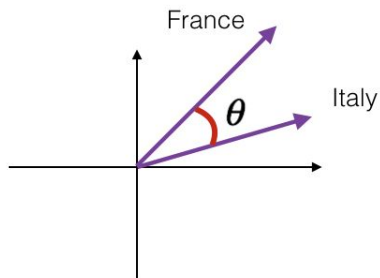
Numericalization

Latest approaches: representation learning

Represent words / sentences / documents as learned vectors which encode their semantic content, trained using machine learning. (word2vec, BERT, GPT, ...)

Use these representations as features for training problem-specific machine learning models.

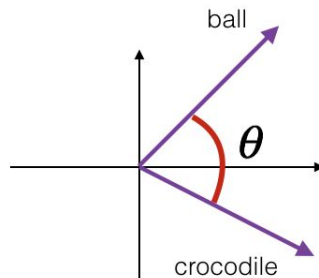
Word similarity: cosine distance



France and Italy are quite similar

θ is close to 0°

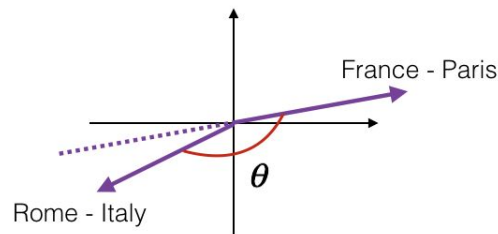
$\cos(\theta) \approx 1$



ball and crocodile are not similar

θ is close to 90°

$\cos(\theta) \approx 0$



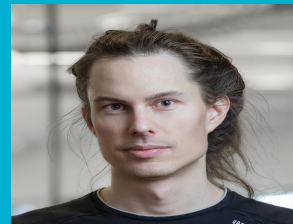
the two vectors are similar but opposite
the first one encodes (city - country)
while the second one encodes (country - city)

θ is close to 180°

$\cos(\theta) \approx -1$

$$\text{similarity}(A, B) = \frac{A \cdot B}{\|A\| \times \|B\|} = \frac{\sum_{i=1}^n A_i \times B_i}{\sqrt{\sum_{i=1}^n A_i^2} \times \sqrt{\sum_{i=1}^n B_i^2}}$$

Dense embeddings you can download!



Word2vec (Mikolov et al., 2013)

<https://code.google.com/archive/p/word2vec/>

Fasttext <http://www.fasttext.cc/> (sub-word information)

Glove (Pennington, Socher, Manning, 2014)

<http://nlp.stanford.edu/projects/glove/>

Embeddings improve everything!
(across NLP tasks)



Word2vec

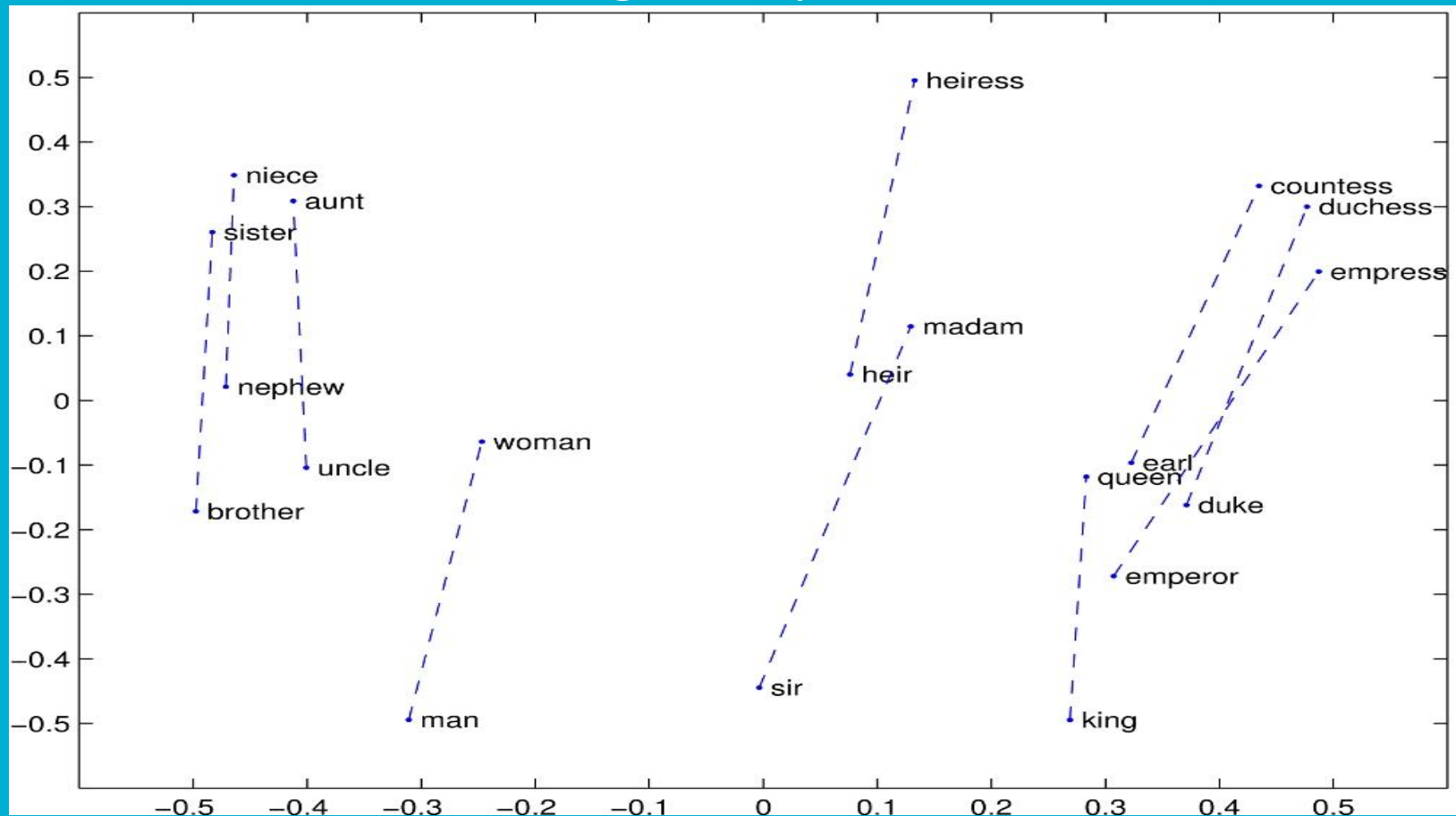
Popular embedding method

Very fast to train

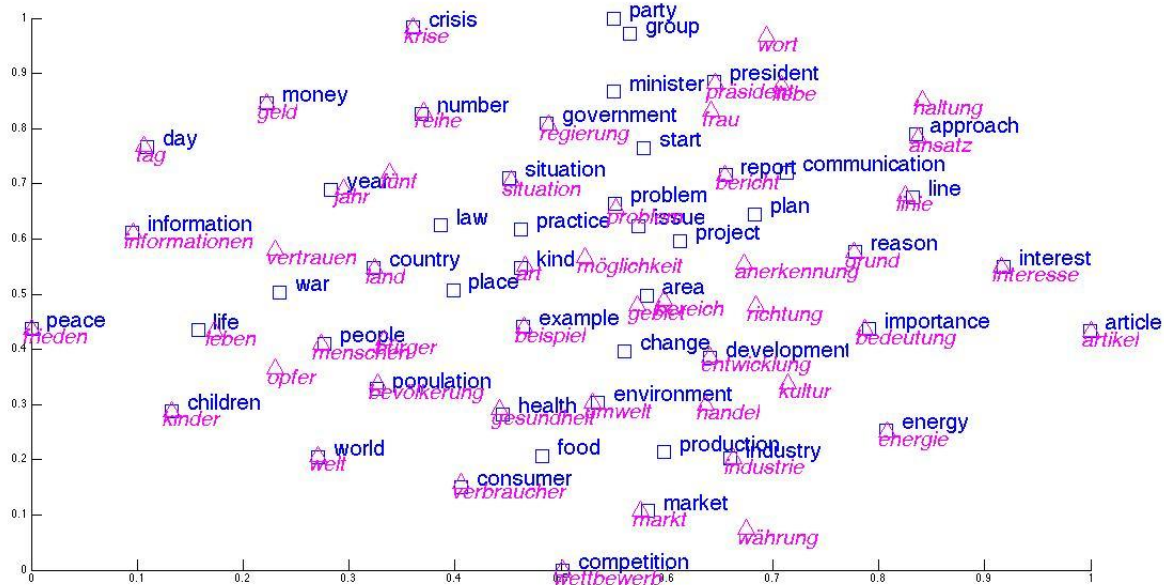
Code available on the web

Idea: **predict** rather than **count**

GloVe visualizations: gender pairs



Multilingual embeddings



History of biased framings of women

Garg, Nikhil, Schiebinger, Londa, Jurafsky, Dan, and Zou, James (2018). Word embeddings quantify 100 years of gender and ethnic stereotypes. *Proceedings of the National Academy of Sciences*, 115(16), E3635–E3644

Embeddings for competence adjectives are biased toward men

◦ *Smart, wise, brilliant, intelligent, resourceful, thoughtful, logical, etc.*

This bias is slowly decreasing

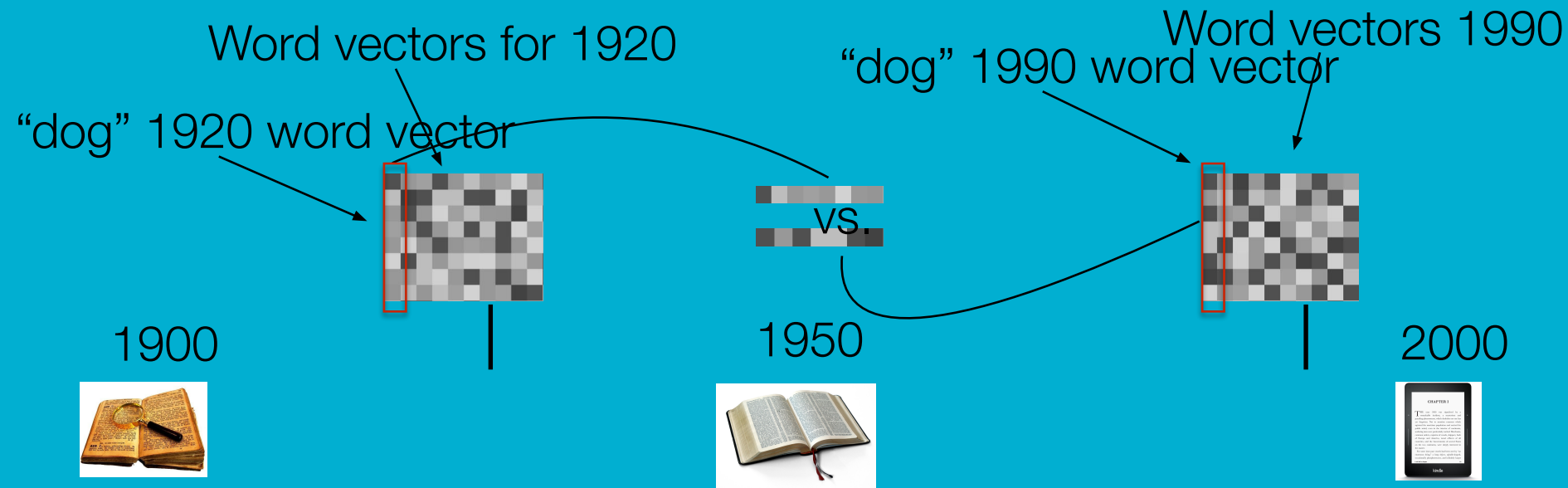
Embeddings can help study word history!

Train embeddings on old books to study changes in word meaning!!



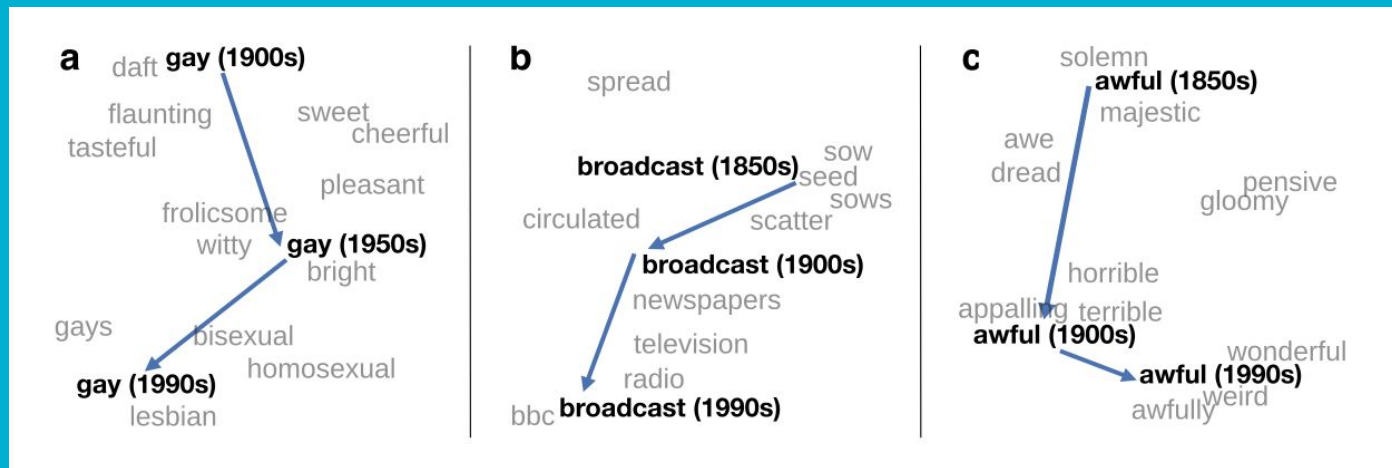
Will Hamilton

Diachronic word embeddings for studying language change!



Visualizing changes

Project 300 dimensions down into 2



~30 million books, 1850-1990, Google Books data

Machine Learning for NLP

Terminologie în învățarea automată

etichetă (label) → what we are trying to predict

caracteristică (feature) → measurable property of a datapoint

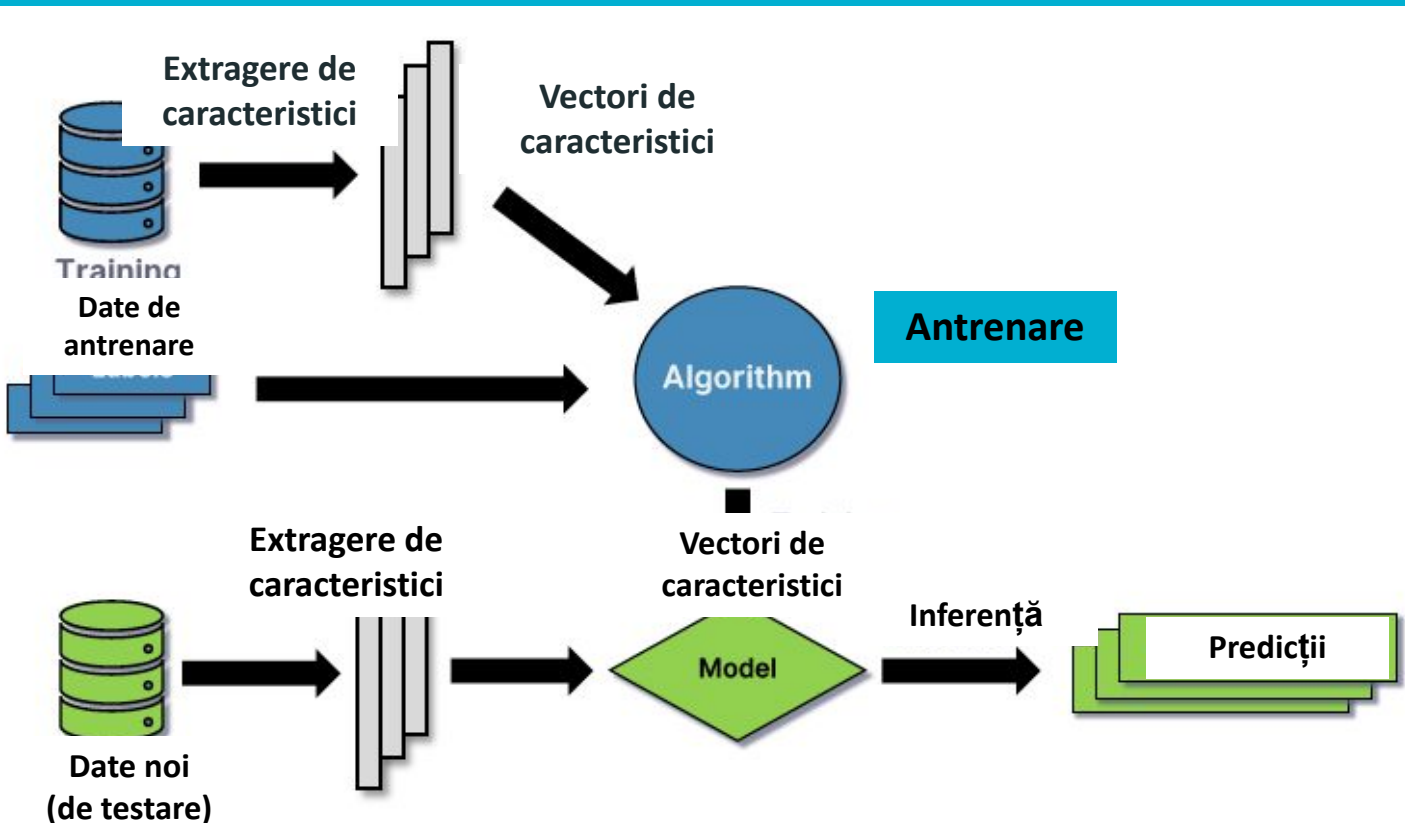
model (ipoteză) → relationship between features and labels

antrenare (training) → establishing the relationship based on a subset of datapoints

inference → doing prediction on unseen data

algorithm defines how training is done, can be restricted to a certain class of hypotheses

Fluxul tipic în învățarea automată



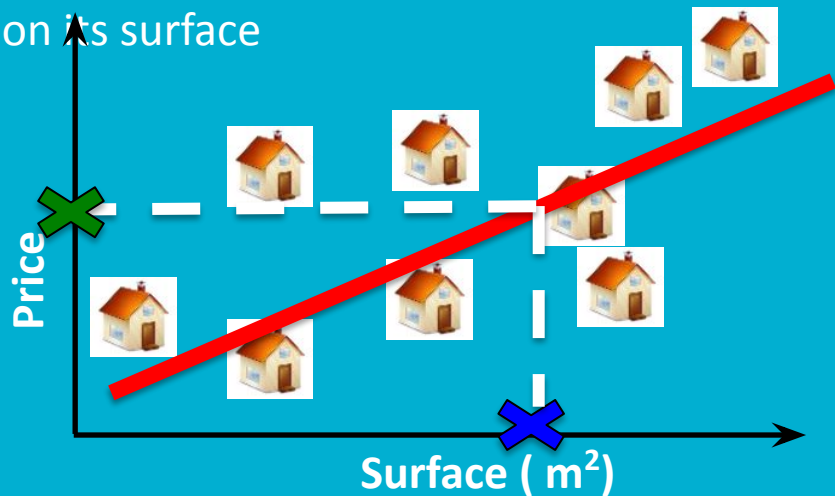
Supervised learning

We want to predict a label for each datapoint

we need annotated data to learn from

Regression: label is a continuous value

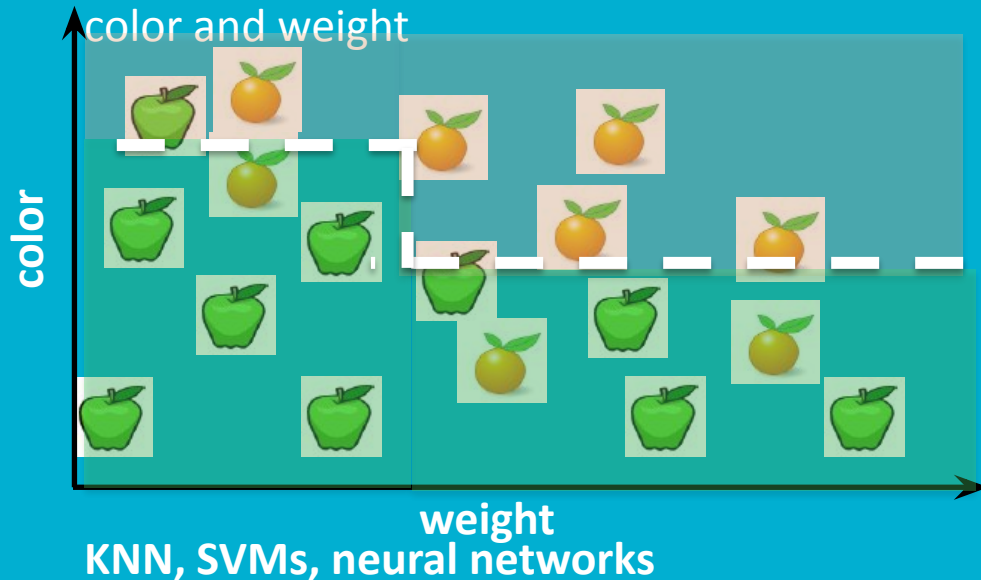
Example: predicting the price of a house based on its surface



Linear regression, ...

Classification: the label takes a discrete value

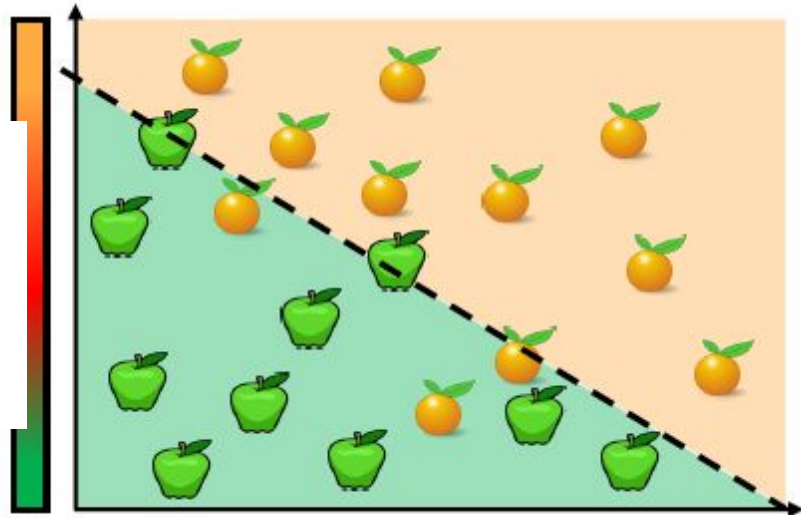
Example: predict what kind of fruit based on color and weight







KNN, SVMs, neural networks

Performance metrics for classification.

Confusion Matrix



Predicted labels

		
 Actual Labels	9	1
	3	8

Accuracy, Precision, Recall

Matrice de confuzie

		Etichete prezise	
		+	-
Etichete reale	+	TP	FN
	-	FP	TN

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + FN + TN}$$

$$\text{Precision} = \frac{TP}{TP + FN}$$

$$\text{Recall} = \frac{TP}{TP + FP}$$

$$F_1 = \frac{2}{\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}} = \frac{2(\text{Precision} * \text{Recall})}{\text{Precision} + \text{Recall}}$$

TP – true positive
FN – false negative
FP – false positive
TN – true negative