

## Content

1	Organisation .....	3
1.1	Topics.....	3
1.2	PÜ & PL.....	3
1.3	Literature .....	3
1.4	Schedule.....	3
2	Basic Text Processing .....	4
2.1	Regular expressions.....	4
2.2	Tokenization.....	4
2.3	Normalization and Stemming.....	5
2.4	Sentence Segmentation and Decision Trees.....	5
2.5	Minimum Edit Distance.....	6
3	Word Senses, Relations & Knowledge Bases .....	7
3.1	Terminology.....	7
3.2	Knowledge Bases .....	7
4	Word Similarity & Sense Disambiguation.....	9
4.1	Similarity & Relatedness Measures.....	9
4.2	Word Sense Disambiguation.....	12
5	Semantic Role Labeling (SRL) .....	13
5.1	Semantic & Thematic Roles .....	13
5.2	Annotated corpora: PropBank .....	13
6	Language Modeling.....	14
6.1	Maximum Likelihood Estimation .....	14
6.2	Perplexity Evaluation.....	16
6.3	Zero Frequency Problem .....	17
6.4	Dealing with huge corpora.....	17
7	Parts of Speech (POS) Tagging.....	18
7.1	CELEX.....	18
7.2	Penn Treebank.....	18

7.3	Brown Corpus .....	19
7.4	Ambiguity.....	19
7.5	POS Tagging Approaches.....	19
8	ML Basics: Metrics for evaluation.....	25
9	Vector Semantics .....	26
9.1	One-hot encoding.....	26
9.2	Distributional semantics.....	26
9.3	Cosine Similarity .....	26
9.4	Term frequency-inverse document frequency (TF-IDF).....	27
9.5	Pointwise mutual information (PMI) .....	28
10	Foundations of Neural Networks.....	29
11	Word Embeddings .....	30
11.1	Word2Vec .....	30
11.2	Global Vectors (GloVe) .....	32
11.3	Fasttext.....	32
11.4	Evaluation.....	32

# 1 Organisation

## 1.1 Topics

1. Introduction
2. Preprocessing
3. Thesauri & Knowledge Bases
4. Word Sense Ambiguation, Entity Linking
5. Language Modeling 1
6. Language Modeling 2
7. POS Tagging 1
8. POS Tagging 2
9. ML Basics: Text Clustering, Classification
10. Vector Semantics 1
11. Vector Semantics 2
12. Neural Networks Introduction
13. Word Embeddings: Static

## 1.2 PÜ & PL

- 4 Home Assignments; not graded but need to be passed (10 of 20 Points total)
- Teams of 3-4 Students allowed
- Exam is 100 points (minimum passing is 50p); 14<sup>th</sup> December

## 1.3 Literature

- Manning & Schütze, Foundations of Statistical NLP, MIT Press 1999
- Jurafsky & Martin, Speech and Language Processing, Prentice Hall 2008
- Eisenstein, An introduction to NLP, Online Draft 2019

## 1.4 Schedule

#	published	due date
<b>Assignment 1</b>	20.09.2022	04.10.2022, 23:59
<b>Assignment 2</b>	11.10.2022	25.10.2022, 23:59
<b>Assignment 3</b>	01.11.2022	15.11.2022, 23:59
<b>Assignment 4</b>	22.11.2022	29.11.2022, 23:59

## 2 Basic Text Processing

### 2.1 Regular expressions

- Formal language for specifying text strings
- Based on fixing 2 kinds of errors
  - Matching strings we shouldn't have (False Positives)
  - Not matching strings we should have matched (False Negatives)
- Regex play a surprisingly large role
  - Sophisticated regex sequences are often the first model for any processing text
  - Regex are used as features for ML Classifiers

### 2.2 Tokenization

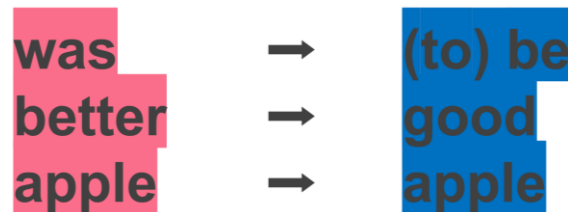
💡 **Lemmatization:** The process of grouping together the inflected forms of a word so they can be analyzed as a single item, identified by the word's lemma, or dictionary form.

📄 **Lemma:** same stem, part of speech, rough word sense

- Cat and cats → same lemma
- Am, are, is → be
- Car, cars, car's, cars' → car

📄 **Wordform:** the full inflected surface form

- Cat and cats → different word forms
- Am, are, is → different word forms



**Example:** "They lay back on the San Francisco grass and looked at the stars and their ..."


📄 **Type:** an element of the vocabulary; above example: 13-11 types → vocabulary ( $V$ )

📄 **Token:** an instance of that type in running text; above example: 15-14 tokens → Tokenset ( $N$ )

#### Issues in Tokenization:

- Apostrophes → Finland's (Finland)
- Abbreviations → I'm (I am), PHD (...)
- Hyphens → Hewlett-Packard, state-of-the-art
- Language differences → compounds vs. segmented nouns (in german)

## 2.3 Normalization and Stemming

 The goal of both stemming and lemmatization is to reduce inflectional forms and sometimes derivationally related forms of a word to a common base form. *Stemming* usually refers to a crude heuristic process that chops off the ends of words in the hope of achieving this goal correctly most of the time, and often includes the removal of derivational affixes. *Lemmatization* usually refers to doing things properly with the use of a vocabulary and morphological analysis of words, normally aiming to remove inflectional endings only and to return the base or dictionary form of a word, which is known as the lemma.

### 2.3.1 Stemming

Techniques to find and reduce terms to the *stem of a word*. Some stemming techniques:

Look-up table


- Problem: building the table

Suffix-stripping algorithms

- If a word ends in “es”, drop the “s” (juices → juice)
- If a word ends with a consonant other than “s”, followed by an “s”, then delete “s” (trains → train; stress → stress)

Stochastic algorithms

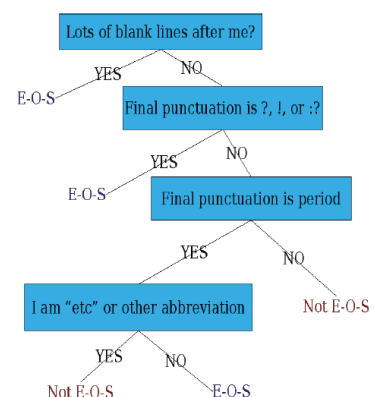
- Learn from annotated text
- Take the context into account to resolve ambiguity

 Less demanding than lemmatization!

## 2.4 Sentence Segmentation and Decision Trees

Sentence segmentations is about identifying borders of sentences.

- “?” and “!” are relatively unambiguous
- “.” is quite ambiguous
  - Sentence boundary
  - Abbreviations (Dr., Inc., ...)
  - Numbers (.2%)



## 2.5 Minimum Edit Distance

Tries to determine how similar two strings are. Is computed by the minimum number of editing operations (insertion, deletion, substitute) that are needed to transform one string into the other. Certain operations can be weighted to increase/decrease distance.

💡 When searching for a path (sequence of edits) from the start string to the final string, we try to minimize the path cost (number of edits).

### Levenshtein Distance

	#	O	D	D
#	0	← 1	← 2	← 3
C	↑ 1	↖ 2	↖ 3	↖ 4
O	↑ 2	↖ 1	← 2	← 3
L	↑ 3	↖ 2	↖ 3	↖ 4
D	↑ 4	↖ 3	← 2	↖ 3

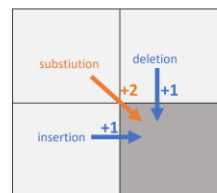
Result: The minimum edit distance is 3.

Alignment:

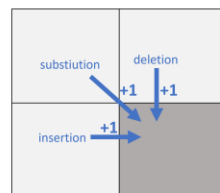
*	O	D	D
C	O	L	D
↑1		↖2	

1. Initialize Matrix with empty string values and add back-tracing arrows.
2. Go over each cell and look at the adjacent cells' value.

Levenshtein: Substitution has cost of 2




Regular: Each operation has cost of 1




3. Pick the lowest value + operation costs for the new cell and add back-tracing arrows to each possible source-cell. **!** If the letters match, copy the diagonal value *without* additional cost.
4. Backtrace the arrows and respective operations to get the final string alignment.

## 3 Word Senses, Relations & Knowledge Bases

### 3.1 Terminology

 *Sense* (or word sense) is a discrete representation of an aspect of a word's meaning.

 *Homonyms* are words that share a form but have unrelated, distinct meanings. For example:

- bank – a financial institution, a sloping land (riverbank)
- bat – club for hitting a ball, nocturnal flying mammal

Homographs:


bank/bank & bat/bat


The “zeugma” test: Two senses of **serve**?


- Which flights **serve** breakfast?
- Does Lufthansa **serve** Philadelphia?
- ¿Does Lufthansa serve breakfast and Philadelphia?

Homophones:


Write/right & piece/peace

 Causes problems for NLP applications, in for example, information retrieval, machine translation or text-to-speech.

 A *polysemous* word has related meaning. “Bank” or “university” may refer to the financial institution as well as the building belonging to it. There's a systematic relationship between organizations and buildings (*metonymy*).

 Antonyms can define a binary opposition or be at opposite ends of a scale (long/short, fast/slow) or be reversive (rise/fall). Autoantonyms bear opposite meanings in one word (fast: moving quickly & fixed firmly in place).

 The part-whole relation: “A leg is part of a chair; a wheel is part of a car.”. Wheel is a *meronym* of car, and car is a *holonym* of wheel.

 One sense is a *hyponym* of another if the first sense is more specific, denoting a subclass of the other (car is a hyponym of vehicle, mango is a hyponym of fruit). Conversely *hypernym*/subordinate (vehicle is a hypernym of car, fruit is a hypernym of mango). Entailment: Sense A is a hyponym of sense B if being an A entails being a B.

### 3.2 Knowledge Bases

Technology used to store complex structured and unstructured information.

#### 3.2.1 WordNet

- A hierarchically organized lexical database
- The *synset* (synonym set) is a set of near synonyms that instantiates a sense or a concept

#### Noun

- *S: (n) bass* (the lowest part of the musical range)
- *S: (n) bass, bass part* (the lowest part in polyphonic music)
- *S: (n) bass, basso* (an adult male singer with the lowest voice)
- *S: (n) sea bass, bass* (the lean flesh of a saltwater fish of the family Serranidae)
- *S: (n) freshwater bass, bass* (any of various North American freshwater fish with lean flesh (especially of the genus Micropterus))
- *S: (n) bass, bass voice, basso* (the lowest adult male singing voice)
- *S: (n) bass* (the member with the lowest range of a family of musical instruments)
- *S: (n) bass* (nontechnical name for any of numerous edible marine and freshwater spiny-finned fishes)

#### Adjective

- *S: (adj) bass, deep* (having or denoting a low vocal or instrumental range) “a deep voice”; “a bass voice is lower than a baritone voice”; “a bass clarinet”

## WordNet Hypernym Hierarchy for “bass”

- **S: (n) bass, basso** (an adult male singer with the lowest voice)
  - **direct hypernym** / **inherited hypernym** / **sister term**
    - **S: (n) singer, vocalist, vocalizer, vocaliser** (a person who sings)
      - **S: (n) musician, instrumentalist, player** (someone who plays a musical instrument (as a profession))
        - **S: (n) performer, performing artist** (an entertainer who performs a dramatic or musical work for an audience)
          - **S: (n) entertainer** (a person who tries to please or amuse)
            - **S: (n) person, individual, someone, somebody, mortal, soul** (a human being) *“there was too much for one person to do”*
              - **S: (n) organism, being** (a living thing that has (or can develop) the ability to act or function independently)
                - **S: (n) living thing, animate thing** (a living (or once living) entity)
                  - **S: (n) whole, unit** (an assemblage of parts that is regarded as a single entity) *“how big is that part compared to the whole?”; “the team is a unit”*
                    - **S: (n) object, physical object** (a tangible and visible entity; an entity that can cast a shadow) *“it was full of rackets, balls and other objects”*
                      - **S: (n) physical entity** (an entity that has physical existence)
                        - **S: (n) entity** (that which is perceived or known or inferred to have its own distinct existence (living or nonliving))

### 3.2.2 ConceptNet

A semantic network consisting of concepts and relations between concepts. Commonsense knowledge in ConceptNet encompasses the spatial, physical, social, temporal, and psychological aspects of everyday life. It represents commonsense knowledge, which is knowledge that every person is assumed to possess. Commonsense knowledge is typically omitted from social communications.

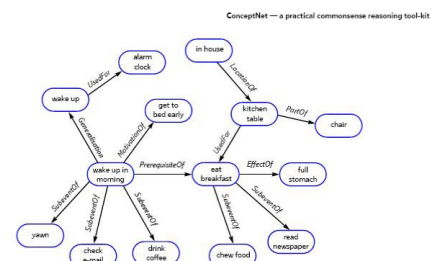



Fig 1. An excerpt from ConceptNet's semantic network of commonsense knowledge. Compound (as opposed to simple) concepts are represented in semi-structured English by composing a verb (e.g. 'drink') with a noun phrase ('coffee') or a prepositional phrase ('in morning').

### 3.2.3 FrameNet

Lexical resource for English, based on frame semantics and supported by corpus evidence. The aim is to document the range of semantics and syntactic combinatory possibilities of each word in each of its senses, through annotation of example sentences and automatic tabulation and display of the annotation results.

 **Semantic frames** are schematic representations of situation types (eating, spying, removing, classifying, etc.) together with lists of the kinds of participants, props, and other conceptual roles that are seen as components of such situations.



## 4 Word Similarity & Sense Disambiguation

### 4.1 Similarity & Relatedness Measures

Two words are more similar if they share more features of meaning. The word *bank* is not similar to the word *slope* but the sense *bank*<sub>1</sub> is similar to *fund*<sub>3</sub> and to *slope*<sub>5</sub>. Similarity can be computed over both words and senses.

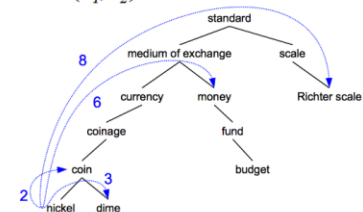
💡 *Car* and *bicycle* are **similar** while *car* and *gasoline* are **related**.

#### 4.1.1 Path-based similarity

Two concepts (senses/synsets) are considered similar if they are near each other in the thesaurus hierarchy (have a short path between them).

📄  $\text{pathlen}(c_1, c_2) = 1 + \text{number of edges in the shortest path in the hypernym graph between sense nodes } c_1 \text{ and } c_2$ . To scale the distances in a range between 0 and 1 we can use:

$$\text{simpath}(c_1, c_2) = 1 / \text{pathlen}(c_1, c_2)$$



$\text{simpath}(\text{nickel}, \text{coin}) = 1/2 = .5$   
 $\text{simpath}(\text{fund}, \text{budget}) = 1/2 = .5$   
 $\text{simpath}(\text{nickel}, \text{currency}) = 1/4 = .25$   
 $\text{simpath}(\text{nickel}, \text{money}) = 1/6 = .17$   
 $\text{simpath}(\text{coinage}, \text{Richter scale}) = 1/6 = .17$

$$\text{simpath}(c_1, c_2) = \frac{1}{\text{pathlen}(c_1, c_2)}$$

❗ Problem: Assumes each link represents a uniform distance. Words that are connected only through very abstract nodes should be considered less similar.

#### 4.1.2 Information content

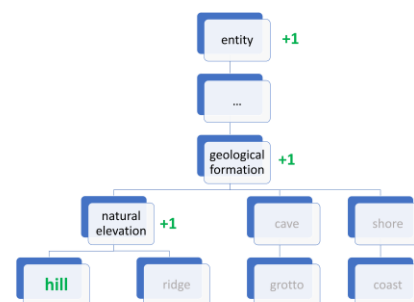
📄  $P(c)$  is defined as the probability that a randomly selected word in a corpus is an instance of concept  $c$ . All words are members of the root node (*entity*), so  $P(\text{root}) = 1$  (as every word that can be randomly selected is inherently an instance of *root*). The lower a node is in hierarchy, the lower its probability.

The probabilities can be obtained by counting the words in a corpus and use the frequencies:

$$P(c) = \frac{\sum_{w \in \text{words}(c)} \text{count}(w)}{N}$$

$\text{words}(c)$  is the set of words that are children of node  $c$ . E.g.,  $\text{words}(\text{natural elevation}) = \{\text{hill}, \text{ridge}\}$ . Each instance of a word counts towards the frequency of its parents.

💡 This allows us to penalize concepts that are very abstract (higher up in the hierarchy).



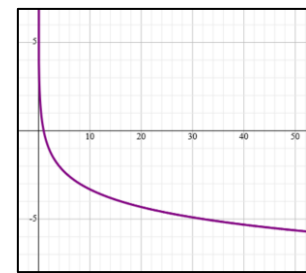
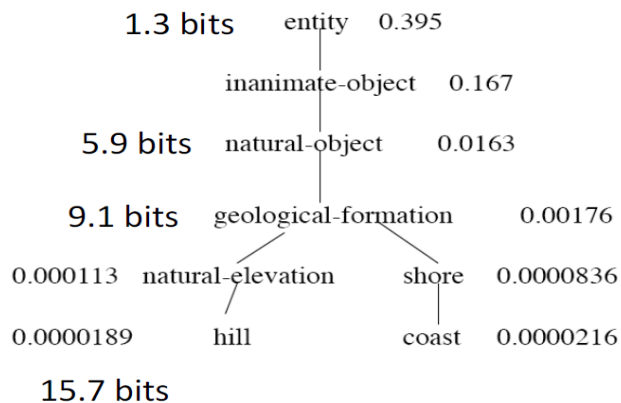
Counting an instance of *hill* in the corpus increases the count of its parents (natural elevation, geo-logical formation and eventually entity) by one.

### 4.1.3 Resnik Method

The *self-information* of an event (*surprisal*) tells us how surprised we are to know it; how much we learn by knowing it.

💡 The more surprising something is, the more it tells us when it happens. We'll measure self-information in *bits*:

$$I(w) = -\log_2 P(w)$$



$$f(x) = -\log_2(x)$$

#### Example: Coin Flip

$P(\text{heads}) = 0,5$  gives us

$$I(\text{heads}) = -\log_2(0,5) = 1 \text{ bit}$$

Using a biased coin does not give us that much information:

$P(\text{heads}) = 0,8$  gives us

$$I(\text{heads}) = -\log_2(0,8) = 0,32 \text{ bits}$$

📄 The most informative node is the lowest common subsumer of two concepts ( $LCS(c_1, c_2)$ ).

$$LCS(\text{hill}, \text{shore}) = \text{geological formation}$$

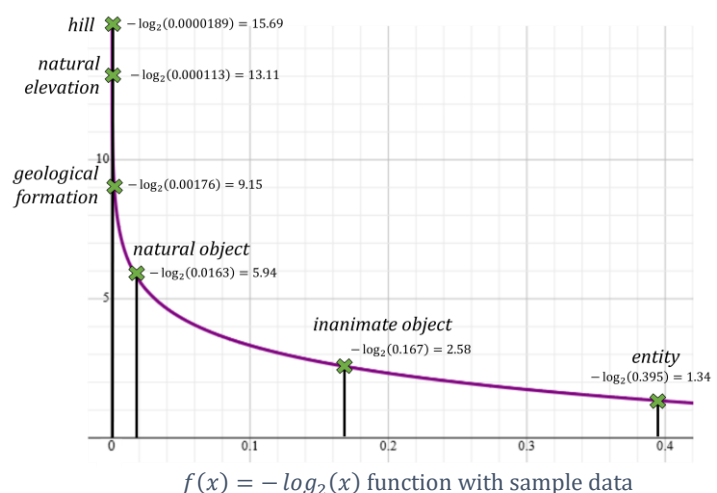
The similarity between two words is related to their common information. The more two words have in common, the more similar they are.

#### Definition

📄 Resnik measures common information as the information content of the most informative (lowest) subsumer (MIS/LCS) of two nodes:

$$\text{sim}_{\text{resnik}}(c_1, c_2) = -\log_2 P(LCS(c_1, c_2))$$

| LCS                  | $P(LCS)$  | $\text{sim}_{\text{resnik}}$ |
|----------------------|-----------|------------------------------|
| entity               | 0,395     | 1,34                         |
| inanimate object     | 0,167     | 2,58                         |
| natural object       | 0,0163    | 5,94                         |
| geological formation | 0,00176   | 9,15                         |
| natural elevation    | 0,000113  | 13,11                        |
| hill                 | 0,0000189 | 15,69                        |



## 4.1.4 Dekang Lin Method

💡 Similarity between sense  $c_1$  and  $c_2$  is not just what they have in common, the more differences  $c_1$  and  $c_2$  have, the less similar they are as well.

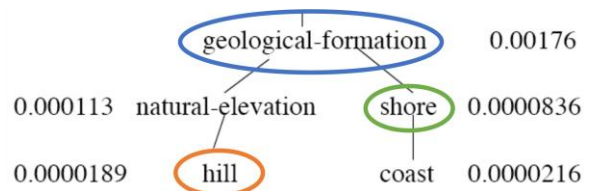
📄 The similarity between  $c_1$  and  $c_2$  is measured by the ratio between the amount of information needed to state the commonality of  $c_1$  and  $c_2$  and the information needed to fully describe what  $c_1$  and  $c_2$  are.

$$sim_{Lin}(c_1, c_2) = \frac{IC(common(c_1, c_2))}{IC(description(c_1, c_2))} = \frac{2 * \log_2 P(LCS(c_1, c_2))}{\log_2 P(c_1) + \log_2 P(c_2)}$$

💡 This describes the IC of the LCS in relation to how far the LCS is away from  $c_1$  and  $c_2$ .

### Example

$$\begin{aligned}
 & sim_{Lin}(shore, hill) \\
 &= \frac{2 * \log_2 P(LCS(shore, hill))}{\log_2 P(shore) + \log_2 P(hill)} \\
 &= \frac{2 * \log_2 P(\text{geological formation})}{\log_2(0.0000189) + \log_2(0.00008236)} \\
 &= \frac{2 * \log_2(0.00176)}{\log_2(0.0000189) + \log_2(0.00008236)} \\
 &= \frac{-18.30042}{-15.69125 + (-13.56770)} \\
 &= 0.63
 \end{aligned}$$



## 4.1.5 Lesk Algorithm

📄 A thesaurus-based measure that looks at glosses (definition of words in thesauri) to determine similarity. Two concepts are regarded similar if their glosses contain similar words.

For example:

- Drawing paper: **paper** that is **specialy prepared** for use in drafting
- Decal: the art of transferring designs from **specialy prepared paper** to wood or glass or metal surface.

💡 For each  $n$ -word *phrase* that's in both glosses, add a score of  $n^2$ :  $1^2 + 2^2 = 5$ .


## 4.2 Word Sense Disambiguation

Given a word in context and a fixed inventory of potential word senses, decide which sense of the word this is.

### 4.2.1 Supervised Machine Learning Methods

A training corpus of words, *tagged* in context with their *sense* that is used to train a classifier that can tag words in unseen text. Needed:

- The tag set (sense inventory)
- The training corpus
- A set of features extracted from the training corpus
- A classifier

 A *tag* refers to an instance of a word in a text (for example: *bass* has 8 possible tags/labels).

```
<wf pos=PRP>He</wf>
<wf pos=VB lemma=recognize wnsn=4 lexs=2:31:00::>recognized</wf>
<wf pos=DT>the</wf>
<wf pos=NN lemma=gesture wnsn=1 lexs=1:04:00::>gesture</wf>
<punc>.</punc>
```

Tagging example from SemCor, a corpus that was manually tagged with WordNet senses.

A simple representation for each observation (instance of a target word) are vectors. Vectors represent, for example, the window of words around the target.

- Collocational: Features about words at *specific positions* near the target word
- Bag-of-words: Features about words that occur *anywhere* in the window

### 4.2.2 Thesaurus/Dictionary Methods

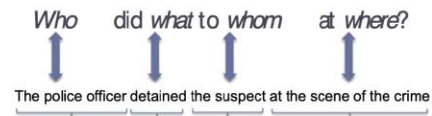
Disambiguate “*bank*” in this sentence. “The bank can guarantee deposits will eventually cover future tuition costs because it invests in adjustable-rate mortgage securities.” given two WordNet senses:

|                   |           |  |
|-------------------|-----------|--|
| bank <sup>1</sup> | Gloss:    | a financial institution that accepts deposits and channels the money into lending activities       |
|                   | Examples: | “he cashed a check at the bank”, “that bank holds the mortgage on my home”                         |
| bank <sup>2</sup> | Gloss:    | sloping land (especially the slope beside a body of water)   |
|                   | Examples: | “they pulled the canoe up on the bank”, “he sat on the bank of the river and watched the currents” |

Now choose the sense with most word overlap between gloss and context: *deposits*, *mortgage* for *bank*<sub>1</sub>. This is called the *simplified Lesk algorithm*.

## 5 Semantic Role Labeling (SRL)

Is about assigning roles to a given NL input.



### 5.1 Semantic & Thematic Roles

Predicates (*bought, sold, purchased*) represent an event. Semantic Roles express the abstract role that arguments of a predicate can take in the event. Neo-Davidsonian event representation:

- *Sasha broke the window.*  $\exists e, x, y \text{ Breaking}(e) \wedge \text{Breaker}(e, \text{Sasha}) \wedge \text{BrokenThing}(e, y) \wedge \text{Window}(y)$
- *Pat opened the door.*  $\exists e, x, y \text{ Opening}(e) \wedge \text{Opener}(e, \text{Pat}) \wedge \text{OpenedThing}(e, y) \wedge \text{Door}(y)$

*Breaker* and *Opener* in this case are volitional actors. There is a direct causal responsibility for their events. They are called **agents**. They act upon will.

*BrokenThing* and *OpenedThing* are **Themes**. They are prototypically inanimate objects affected in some way by the action.

| Thematic Role | Definition  | Example  |
|---------------|---|--|
| AGENT         | The volitional causer of an event                   | <i>The waiter</i> spilled the soup.                                    |
| EXPERIENCER   | The experiencer of an event                         | <i>John</i> has a headache.  |
| FORCE         | The non-volitional causer of the event              | <i>The wind</i> blows debris from the mall into our yards.             |
| THEME         | The participant most directly affected by an event  | Only after Benjamin Franklin broke <i>the ice</i> ...                  |
| RESULT        | The end product of an event                         | The city built a <i>regulation-size baseball diamond</i> ...           |
| CONTENT       | The proposition or content of a propositional event | Mona asked " <i>You met Mary Ann at a supermarket?</i> "               |
| INSTRUMENT    | An instrument used in an event                      | He poached catfish, stunning them <i>with a shocking device</i> ...    |
| BENEFICIARY   | The beneficiary of an event                         | Whenever Ann Callahan makes hotel reservations <i>for her boss</i> ... |
| SOURCE        | The origin of the object of a transfer event        | I flew in <i>from Boston</i> .   |
| GOAL          | The destination of an object of a transfer event    | I drove <i>to Portland</i> .   |

Given these abstract roles, we can annotate sentences with them.

The problem with thematic roles is that it's hard to create a standard set of roles or formally define them. They often need to be fragmented to be defined.

### 5.2 Annotated corpora: PropBank

Proposition Bank (PropBank) it aims to come up with an inventory of very abstract semantic roles as well as an annotated corpus of semantic roles. PropBank Frame Files look like this:

#### agree.01

Arg0: Agreeer

Arg1: Proposition

Arg2: Other entity agreeing

Ex1: [Arg0 The group] *agreed* [Arg1 it wouldn't make an offer].

Ex2: [ArgM-TMP Usually] [Arg0 John] *agrees* [Arg2 with Mary] [Arg1 on everything].

## 6 Language Modeling

? When given a partial sentence, can we devise a method to predict what the next word will be?

This task can be formalized by using N-gram models. N-grams are *token sequences* of length  $N$ . For example: "I notice three guys standing on the ..." contains the following 2-grams: (I notice), (notice three), (three guys), ..., (on the).

💡 We can use the knowledge of the counts of N-grams to assess the conditional probability of candidate words as the next word in a sequence:  $P(w_n | w_1, w_2, \dots, w_{n-1})$

So, for the example sentence:

$P(\text{"I notice three guys standing on the"}) =$   
 $P(\text{"I"}) * P(\text{"notice"} | \text{"I"}) * P(\text{"three"} | \text{"I notice"}) * \dots *$   
 $P(\text{"the"} | \text{"I notice three guys standing on"})$

|   |
|---|
| Product/Chain Rule  |
| $P(A,B) = P(A)P(B A)$   |
| For sequences...  |
| • $P(A,B,C,D) = P(A)P(B A)P(C A,B)P(D A,B,C)$   |
| In general  |
| • $P(x_1, x_2, x_3, \dots, x_n) = P(x_1)P(x_2 x_1)P(x_3 x_1, x_2) \dots P(x_n x_1 \dots x_{n-1})$ |

These probabilities can be obtained by counting through corpora appearances. Problem is that long sequences occur very rarely! Therefore, using  $n$ -grams to only consider the last few words could serve as an approximation for the sentence.

When we use a bigram model to predict the conditional probability of the next word, we are thus making the following approximation:

$$P(w_n | w_1^{n-1}) \approx P(w_n | w_{n-1}) \quad (3.7)$$

! This is called the **independence assumption**, as we assume that the probability of word  $n$  given all its preceding words is roughly the same as the probability of word  $n$  given only word  $n-1$ .

This can be extended to trigrams, 4-grams, 5-grams. But in general, this is an insufficient model of language, because language has long-distance dependencies.

### 6.1 Maximum Likelihood Estimation

📄 The maximum likelihood estimate of some parameter of a model  $M$  from a training set  $T$  is the estimate that maximizes the likelihood of the training set  $T$  given the model  $M$ .

<s> I am Sam </s>

<s> Sam I am </s>

<s> I do not like green eggs and ham </s>

$$P(w_i | w_{i-1}) = \frac{\text{count}(w_{i-1}, w_i)}{\text{count}(w_{i-1})} \quad P(I | <s>) = 2/3 \quad P(\text{Sam} | <s>) = 1/3 \quad P(\text{am} | I) = 2/3$$

Example: Berkley Restaurant Project (is a collection of sentences uttered in restaurants)

Unigram counts (out of 9222 sentences)

| i    | want | to   | eat | chinese | food | lunch | spend |
|------|------|------|-----|---------|------|-------|-------|
| 2533 | 927  | 2417 | 746 | 158     | 1093 | 341   | 278   |

Bigram counts (out of 9222 sentences)

|         | i  | want | to  | eat | chinese | food | lunch | spend |
|---------|----|------|-----|-----|---------|------|-------|-------|
| i       | 5  | 827  | 0   | 9   | 0       | 0    | 0     | 2     |
| want    | 2  | 0    | 608 | 1   | 6       | 6    | 5     | 1     |
| to      | 2  | 0    | 4   | 686 | 2       | 0    | 6     | 211   |
| eat     | 0  | 0    | 2   | 0   | 16      | 2    | 42    | 0     |
| chinese | 1  | 0    | 0   | 0   | 0       | 82   | 1     | 0     |
| food    | 15 | 0    | 15  | 0   | 1       | 4    | 0     | 0     |
| lunch   | 2  | 0    | 0   | 0   | 0       | 1    | 0     | 0     |
| spend   | 1  | 0    | 1   | 0   | 0       | 0    | 0     | 0     |

Normalizing by unigrams gives us

|         | i       | want | to     | eat    | chinese | food   | lunch  | spend   |
|---------|---------|------|--------|--------|---------|--------|--------|---------|
| i       | 0.002   | 0.33 | 0      | 0.0036 | 0       | 0      | 0      | 0.00079 |
| want    | 0.0022  | 0    | 0.66   | 0.0011 | 0.0065  | 0.0065 | 0.0054 | 0.0011  |
| to      | 0.00083 | 0    | 0.0017 | 0.28   | 0.00083 | 0      | 0.0025 | 0.087   |
| eat     | 0       | 0    | 0.0027 | 0      | 0.021   | 0.0027 | 0.056  | 0       |
| chinese | 0.0063  | 0    | 0      | 0      | 0       | 0.52   | 0.0063 | 0       |
| food    | 0.014   | 0    | 0.014  | 0      | 0.00092 | 0.0037 | 0      | 0       |
| lunch   | 0.0059  | 0    | 0      | 0      | 0       | 0.0029 | 0      | 0       |
| spend   | 0.0036  | 0    | 0.0036 | 0      | 0       | 0      | 0      | 0       |



## Example

Given the following corpus of sentences:

- <s> Lea likes frogs </s>
- <s> Kim likes frogs too </s>
- <s> Kim likes apples too </s>

1. Construct a unigram model.

$$P(w) = \frac{\text{count}(w)}{N}$$

# of word/token  $w$  in corpus  
total # of words/tokens in corpus

$$N = |\text{corpus}| = \sum_{w \in V} \text{count}(w) = 17$$

| Unigram | Count | Probability                       |
|---------|-------|-----------------------------------|
| <s>     | 3     | $P(<s>) = \frac{3}{17}$           |
| Lea     | 1     | $P(\text{Lea}) = \frac{1}{17}$    |
| likes   | 3     | $P(\text{likes}) = \frac{3}{17}$  |
| frogs   | 2     | $P(\text{frogs}) = \frac{2}{17}$  |
| Kim     | 2     | $P(\text{Kim}) = \frac{2}{17}$    |
| too     | 2     | $P(\text{too}) = \frac{2}{17}$    |
| apples  | 1     | $P(\text{apples}) = \frac{1}{17}$ |
| </s>    | 3     | $P(</s>) = \frac{3}{17}$          |

$P(<s>\text{Frogs likes apples too } </s>)$

$$= \frac{3}{17} * \frac{2}{17} * \frac{3}{17} * \frac{1}{17} * \frac{2}{17} * \frac{3}{17} \approx 4.4 * 10^{-6}$$

$P(<s>\text{Kim likes Lea } </s>) =$

$$\frac{3}{17} * \frac{2}{17} * \frac{3}{17} * \frac{1}{17} * \frac{3}{17} \approx 0.00004$$

2. Construct a bigram model.

$$P(w_i | w_{i-1}) = \frac{\text{count}(w_{i-1}, w_i)}{\text{count}(w_{i-1})}$$

# of times word  $w_i$  follows word  $w_{i-1}$  in the corpus  
total # of times word  $w_{i-1}$  appears in the corpus

| Bigram       | Count | Probability   |
|--------------|-------|---------------|
| <s> Lea      | 1     | $\frac{1}{3}$ |
| Lea likes    | 1     | $\frac{1}{1}$ |
| likes frogs  | 2     | $\frac{2}{3}$ |
| frogs </s>   | 1     | $\frac{1}{2}$ |
| <s> Kim      | 2     | $\frac{2}{3}$ |
| Kim likes    | 2     | $\frac{2}{2}$ |
| frogs too    | 1     | $\frac{1}{2}$ |
| likes apples | 1     | $\frac{1}{3}$ |
| apples too   | 1     | $\frac{1}{1}$ |
| too </s>     | 2     | $\frac{2}{2}$ |

$P(<s>\text{Lea likes apples too } </s>)$

$$= \frac{1}{3} * \frac{1}{1} * \frac{1}{3} * \frac{1}{1} * \frac{2}{2} \approx \frac{1}{9}$$

## 6.2 Perplexity Evaluation

Evaluates language models *intrinsically*. Measures *how surprising* the prediction of a language model would be. Perplexity is the inverse probability of the test set, normalized by the number of words.

**Unigram**

$$PP(W) = \sqrt[N]{\prod_{i=1}^N \frac{1}{P(w_i)}}$$

**Bigram**

$$PP(W) = \sqrt[N]{\prod_{i=1}^N \frac{1}{P(w_i|w_{i-1})}}$$

*N is the length of the sentence W*

💡 Minimizing perplexity is the same as maximizing probability.

### Example (1/2) - Unigram model perplexity

For reference, see example from 6.1 *Maximum Likelihood Estimation*

$PP(<s>\text{Frogs likes apples too } </s>)$

$$\begin{aligned} &= \sqrt[6]{\prod_{i=1}^6 \frac{1}{P(w_i)}} = \\ &= \sqrt[6]{\frac{1}{\frac{3}{17} * \frac{2}{17} * \frac{3}{17} * \frac{1}{17} * \frac{2}{17} * \frac{3}{17}}} \\ &= \sqrt[6]{\frac{17}{3} * \frac{17}{2} * \frac{17}{3} * \frac{17}{1} * \frac{17}{2} * \frac{17}{3}} \\ &= \sqrt[6]{\frac{17^6}{108}} \approx \mathbf{7,79} \end{aligned}$$

$PP(<s>\text{Kim likes Lea } </s> >)$

$$\begin{aligned} &= \sqrt[5]{\prod_{i=1}^5 \frac{1}{P(w_i)}} = \\ &= \sqrt[5]{\frac{1}{\frac{3}{17} * \frac{2}{17} * \frac{3}{17} * \frac{1}{17} * \frac{3}{17}}} \\ &= \sqrt[5]{\frac{17}{3} * \frac{17}{2} * \frac{17}{3} * \frac{17}{1} * \frac{17}{3}} \\ &= \sqrt[5]{\frac{17^5}{54}} \approx \mathbf{7,76} \end{aligned}$$

### Example (2/2) - Bigram model perplexity


For reference, see example from 6.1 *Maximum Likelihood Estimation*

$$\begin{aligned} PP(<s>\text{Lea likes apples too } </s>) &= \sqrt[5]{\prod_{i=1}^5 \frac{1}{P(w_i|w_{i-1})}} = \sqrt[5]{\frac{1}{\frac{1}{3} * \frac{1}{1} * \frac{1}{3} * \frac{1}{1} * \frac{2}{2}}} = \sqrt[5]{\frac{3}{1} * \frac{1}{1} * \frac{3}{1} * \frac{1}{1} * \frac{2}{2}} \\ &= \sqrt[5]{9} \approx \mathbf{1,55} \end{aligned}$$

💡 The lower the perplexity is, the better the model. The results of the bigram model do not *surprise* as much as the ones from the unigram model – which is logical. A trigram model would have an even lower perplexity.



## 6.3 Zero Frequency Problem

 Bigrams with zero probability (never seen/counted) means that we cannot compute complexity (can't divide by 0). This can be resolved by the “add-one” (Laplace) smoothing, where 1 count is added to every word occurrence:

Unigrams:

$$P_{add-1}(w_i) = \frac{c(w_i) + 1}{N + V}$$

Bigrams:

$$P_{add-1}(w_i|w_{i-1}) = \frac{c(w_{i-1}, w_i) + 1}{c(w_{i-1}) + V}$$

|         | i  | want | to  | eat | chinese | food | lunch | spend |
|---------|----|------|-----|-----|---------|------|-------|-------|
| i       | 6  | 828  | 1   | 10  | 1       | 1    | 1     | 3     |
| want    | 3  | 1    | 609 | 2   | 7       | 7    | 6     | 2     |
| to      | 3  | 1    | 5   | 687 | 3       | 1    | 7     | 212   |
| eat     | 1  | 1    | 3   | 1   | 17      | 3    | 43    | 1     |
| chinese | 2  | 1    | 1   | 1   | 1       | 83   | 2     | 1     |
| food    | 16 | 1    | 16  | 1   | 2       | 5    | 1     | 1     |
| lunch   | 3  | 1    | 1   | 1   | 1       | 2    | 1     | 1     |
| spend   | 2  | 1    | 2   | 1   | 1       | 1    | 1     | 1     |


|         | i       | want    | to      | eat     | chinese | food    | lunch   | spend   |
|---------|---------|---------|---------|---------|---------|---------|---------|---------|
| i       | 0.0015  | 0.21    | 0.00025 | 0.0025  | 0.00025 | 0.00025 | 0.00025 | 0.00075 |
| want    | 0.0013  | 0.00042 | 0.26    | 0.00084 | 0.0029  | 0.0029  | 0.0025  | 0.00084 |
| to      | 0.00078 | 0.00026 | 0.0013  | 0.18    | 0.00078 | 0.00026 | 0.0018  | 0.055   |
| eat     | 0.00046 | 0.00046 | 0.0014  | 0.00046 | 0.0078  | 0.0014  | 0.02    | 0.00046 |
| chinese | 0.0012  | 0.00062 | 0.00062 | 0.00062 | 0.00062 | 0.052   | 0.0012  | 0.00062 |
| food    | 0.0063  | 0.00039 | 0.0063  | 0.00039 | 0.00079 | 0.002   | 0.00039 | 0.00039 |
| lunch   | 0.0017  | 0.00056 | 0.00056 | 0.00056 | 0.00056 | 0.0011  | 0.00056 | 0.00056 |
| spend   | 0.0012  | 0.00058 | 0.0012  | 0.00058 | 0.00058 | 0.00058 | 0.00058 | 0.00058 |

! This is just a blunt instrument, and we'll investigate better methods.

## 6.4 Dealing with huge corpora

- Only store N-grams with count > threshold
- Remove singletons of higher-order n-grams
- Use efficient data structures
- ....

## 7 Parts of Speech (POS) Tagging

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

The koala put the keys on the table  
 ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑  
 DET N V DET N P DET N

POS Examples:

- N – noun – chair, bandwidth
- V – verb – sit, study
- ADJ – adjective – purple, tall
- ADV – adverb – slowly
- P – preposition – of, by, to
- PRO – pronoun – I, me, mine
- DET – determiner – the, a, that

 POS is useful for speech synthesis, information extraction, machine translation and others.

### Open Word Classes

- A small, fixed membership of words in these classes
- Propositions, auxiliaries, pronouns
- Usually function words (short common words which play a role in grammar)

### Closed Word Classes

- New words can be added to the classes all the time
- Nouns, Verbs, Adjectives, Adverbs

## 7.1 CELEX

A database comprising of phonological, morphological, syntactic and frequency properties of tokens. Contains 17,9 million tokens.

**Please select a database:**

|   |                                     |                                    |  |
|---|-------------------------------------|------------------------------------|--|
| <b>Dutch:</b>                           | <b>English:</b>                     | <b>German:</b>                     | <b>Cyrillic:</b>                           |
| <a href="#">Dutch Abbreviations</a>     | <a href="#">English Corpustypes</a> | <a href="#">German Corpustypes</a> | <a href="#">German-Xakas-Tuvan Lexicon</a> |
| <a href="#">Dutch Corpustypes</a>       | <a href="#">English Lemmas</a>      | <a href="#">German Lemmas</a>      | <a href="#">English-Tuvan Lexicon</a>      |
| <a href="#">Dutch Lemmas</a>            | <a href="#">English Syllables</a>   | <a href="#">German Syllables</a>   |  |
| <a href="#">Dutch New Abbreviations</a> | <a href="#">English Wordforms</a>   | <a href="#">German Wordforms</a>   |  |
| <a href="#">Dutch Syllables</a>         |                                     |                                    |  |
| <a href="#">Dutch Uitsidenboogaart</a>  |                                     |                                    |  |
| <a href="#">Dutch Wordforms</a>         |                                     |                                    |  |

## 7.2 Penn Treebank

The presumably first large syntactically annotated corpus. It is annotated with POS information and skeletal syntactic structure. It contains over 4,5 million words.

The “Penn TreeBank tagset” contains a very fine-grained set of tags (45 tags).

**Table 4:  
Penn Treebank  
(as of 11/92)**

| Description                    | Tagged for<br>Part-of-Speech<br>(Tokens) | Skeletal<br>Parsing<br>(Tokens) |
|--------------------------------|--|---------------------------------|
| Dept. of Energy abstracts      | 231,404                                  | 231,404                         |
| Dow Jones Newswire stories     | 3,065,776                                | 1,061,166                       |
| Dept. of Agriculture bulletins | 78,555                                   | 78,555                          |
| Library of America texts       | 105,652                                  | 105,652                         |
| MUC-3 messages                 | 111,828                                  | 111,828                         |
| IBM Manual sentences           | 89,121                                   | 89,121                          |
| WBUR radio transcripts         | 11,589                                   | 11,589                          |
| ATIS sentences                 | 19,832                                   | 19,832                          |
| Brown Corpus, retagged         | 1,172,041                                | 1,172,041                       |
| <b>Total:</b>                  | <b>4,885,798</b>                         | <b>2,881,188</b>                |

## 7.3 Brown Corpus

Contains 1 million words of American English texts, sampled from 15 different text categories. It is divided into 500 samples of 2000+ words each. They represent a wide range of styles and varieties of prose. It does not include the “brown Corpus Tagset”!

## 7.4 Ambiguity

Words often have more than one POS, for example: *back* → back door (JJ), my back (NN), ...

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

## 7.5 POS Tagging Approaches

### 7.5.1 Rule-based: ENGTWOL

Abstract approach

1. Start with a dictionary (*back* – NN, JJ, RB, VB; *bill* – NN, VB; ...)
2. Assign all possible tags to words from the dictionary
3. Write rules by hand to selectively remove tags
4. Leaving the correct tag for each word

💡 This approach works and produces accurate results, but it is extremely labor-intensive!

## 7.5.2 Stochastic: Hidden Markov Models (HMMs)

Stochastic taggers use a training corpus to compute probability of a tag in a context. Using an HMM to do POS tagging is a special case of Bayesian inference. We are given a sentence (an *observation* or *sequence of observations*): "Secretariat is expected to start tomorrow"

? What is the best sequence of tags that corresponds to this sequence of observations?

Probabilistic view: Consider all possible sequences of tags, out of this universe of sequences, choose the tag sequence which is most probable give the observation sequence of  $n$  words  $w_1 \dots w_n$ .

$$\hat{t}_{1:n} = \operatorname{argmax}_{t_{1:n}} P(t_{1:n} | w_{1:n})$$

- The roof symbol " $\hat{\phantom{x}}$ " in  $\hat{t}_{1:n}$  means "our estimate of the best one"
- The expression  $\operatorname{argmax}_{t_{1:n}} f(t)$  refers to the maximum of  $f(t)$  for the values 1 to  $n$ .

💡 Use Bayes rule to transform this equation into a set of other probabilities that are easier to compute:

$$\hat{t}_{1:n} = \operatorname{argmax}_{t_{1:n}} P(t_{1:n} | w_{1:n}) \quad \downarrow \text{by using the bayes rule for conditional probability: } P(X, Y) = \frac{P(Y|X) * P(X)}{P(Y)}$$

$$= \operatorname{argmax}_{t_{1:n}} \frac{P(t_{1:n} | w_{1:n}) * P(t_{1:n})}{\underbrace{P(w_{1:n})}}$$

Can be neglected as it does not change for the different values of  $t$  and therefore does not matter in the context of the  $\operatorname{argmax}()$ -function

$$= \operatorname{argmax}_{t_{1:n}} \underbrace{P(t_{1:n} | w_{1:n})}_{\text{likelihood}} * \underbrace{P(t_{1:n})}_{\text{prior}}$$

likelihood

prior

$$\approx \prod_{i=1}^n P(w_i | t_i)$$

$$\approx \prod_{i=1}^n P(t_i | t_{i-1})$$

Product of each word given its tag

Product of each tag given the tag that occurred before

$$\approx \operatorname{argmax}_{t_{1:n}} \prod_{i=1}^n P(w_i | t_i) * P(t_i | t_{i-1})$$

$$\hat{t}_{1:n} \approx \underset{t_{1:n}}{\operatorname{argmax}} \prod_{i=1}^n \underbrace{P(w_i|t_i)}_{\text{word likelihood probabilities}} * \underbrace{P(t_i|t_{i-1})}_{\text{tag transition probabilities}}$$

What is the probability that word  $w_i$  occurs given the tag  $t_i$ ?

💡 This is computed by counting in labeled corpus!

$$P(w_i|t_i) = \frac{\text{count}(w_i, t_i)}{\text{count}(t_i)}$$

or for example:

$$P(\text{"is"}|\text{VBZ}) = \frac{10.073}{21.627} = 0,47$$

What is the probability that tag  $t_i$  occurs given the previous tag  $t_{i-1}$ ?

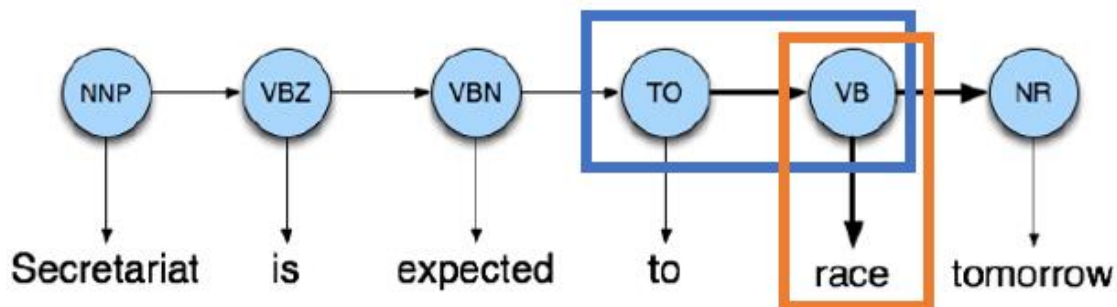
💡 This is computed by counting in labeled corpus!

$$P(t_i|t_{i-1}) = \frac{\text{count}(t_i, t_{i-1})}{\text{count}(t_{i-1})}$$

or for example:

$$P(\text{NN}|\text{DT}) = \frac{56.509}{116.454} = 0,49$$

$$P(t_i|t_{i-1}) = P(\text{VB}|\text{TO}) = 0,83$$



$$P(w_i|t_i) = P(\text{"race"}|\text{VB}) = 0,00012$$

## 7.5.3 Viterbi-Algorithm

Using the Viterbi algorithm, determine the most likely sequence of states (i.e., sequence of POS tags) for the observation sequence: THE FANS WATCH THE RACE

| State | $\Pi_{state}$ |         |    |     |     |     |    |       |      |
|-------|---------------|---------|----|-----|-----|-----|----|-------|------|
| DT    | 0.8           |         | A  | DT  | N   | V   | B  | THE   | FANS |
| N     | 0.2           | $\pi_N$ | DT | 0   | 0.9 | 0.1 | DT | 0.2   | 0    |
| V     | 0             |         | N  | 0   | 0.5 | 0.5 | N  | 0     | 0.1  |
|       |               |         | V  | 0.5 | 0.5 | 0   | V  | 0     | 0.2  |
|       |               |         |    |     |     |     |    | WATCH | RACE |
|       |               |         |    |     |     |     |    |       | 0.3  |

Initial probabilities  $\pi$

Transition probabilities  $a$

Emission probabilities  $b$

$a_{N \rightarrow V}$

$b_N(RACE)$

$$v_t(i) = \max_{s \in \text{states}} \underbrace{v_{t-1}(s)}_{\text{previous viterbi path probability}} * \underbrace{a_{s \rightarrow i}}_{\text{transition probability}} * \underbrace{b_i(o_t)}_{\text{emission probability}}$$

THE  
DT  
0,16  
~~N~~  
~~V~~

**Position t=1** word = "THE"

$$v_1(DT) = \pi_{DT} * b_{DT}(THE) = 0,8 * 0,2 = 0,16$$

$$v_1(N) = \pi_N * b_N(THE) = 0,2 * 0 = 0$$

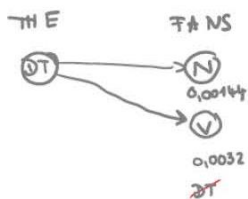
$$v_1(V) = \pi_V * b_V(THE) = 0 * 0 = 0$$

**Position t=2** word = "FANS"

$$\begin{aligned} v_2(DT) &= \max(v_1(DT) * a_{DT \rightarrow DT}; v_1(N) * a_{N \rightarrow DT}; v_1(V) * a_{V \rightarrow DT}) * b_{DT}(FANS) \\ &= \max(0,16 * 0; 0; 0) * 0 \\ &= 0 \rightarrow \psi_2(DT) = \emptyset \end{aligned}$$

$$\begin{aligned} v_2(N) &= \max(v_1(DT) * a_{DT \rightarrow N}; v_1(N) * a_{N \rightarrow N}; v_1(V) * a_{V \rightarrow N}) * b_N(FANS) \\ &= \max(0,16 * 0,9; 0; 0) * 0,1 \\ &= 0,0144 \rightarrow \psi_2(N) = DT \end{aligned}$$

$$\begin{aligned} v_2(V) &= \max(v_1(DT) * a_{DT \rightarrow V}; v_1(N) * a_{N \rightarrow V}; v_1(V) * a_{V \rightarrow V}) * b_V(FANS) \\ &= \max(0,16 * 0,1; 0; 0) * 0,2 \\ &= 0,0032 \rightarrow \psi_2(V) = DT \end{aligned}$$

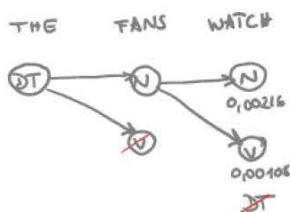


**Position t=3** word = "WATCH"

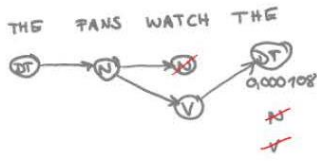
$$\begin{aligned} v_3(DT) &= \max(v_2(DT) * a_{DT \rightarrow DT}; v_2(N) * a_{N \rightarrow DT}; v_2(V) * a_{V \rightarrow DT}) * b_{DT}(WATCH) \\ &= \max(\dots) * 0 = 0 \rightarrow \psi_3(DT) = \emptyset \end{aligned}$$

$$\begin{aligned} v_3(N) &= \max(v_2(DT) * a_{DT \rightarrow N}; v_2(N) * a_{N \rightarrow N}; v_2(V) * a_{V \rightarrow N}) * b_N(WATCH) \\ &= \max(0; 0,0144 * 0,5; 0,0032 * 0,5) * 0,3 \\ &= 0,0072 * 0,3 = 0,00216 \rightarrow \psi_3(N) = N \end{aligned}$$

$$\begin{aligned} v_3(V) &= \max(v_2(DT) * a_{DT \rightarrow V}; v_2(N) * a_{N \rightarrow V}; v_2(V) * a_{V \rightarrow V}) * b_V(WATCH) \\ &= \max(0; 0,0144 * 0,5; 0,0032 * 0,5) * 0,15 \\ &= 0,0072 * 0,15 = 0,00108 \rightarrow \psi_3(V) = N \end{aligned}$$



**Position  $t=4$**  word = "THE"



$$\begin{aligned} v_3(DT) &= \max(v_3(DT) * a_{DT \rightarrow DT}; v_3(N) * a_{N \rightarrow DT}; v_3(V) * a_{V \rightarrow DT}) * b_{DT}(THE) \\ &= \max(0; 0; 0,00108 * 0,5) * 0,2 \\ &= 0,000108 \end{aligned} \quad \rightarrow \psi_4(DT) = V$$

$$v_3(N) = \max(\dots) * \overbrace{b_N(THE)}^{=0} = 0 \quad \rightarrow \psi_4(N) = \emptyset$$

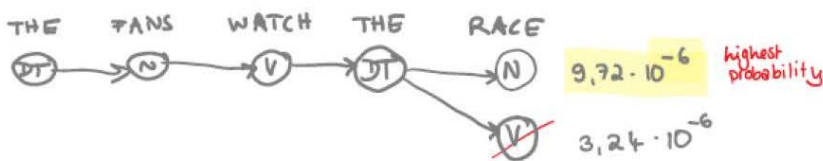
$$v_3(V) = \max(\dots) * \overbrace{b_V(THE)}^{=0} = 0 \quad \rightarrow \psi_4(V) = \emptyset$$

**Position  $t=5$**  word = "RACE"

$$v_5(DT) = \max(\dots) * \overbrace{b_{DT}(RACE)}^{=0} = 0 \quad \rightarrow \psi_5(DT) = \emptyset$$

$$\begin{aligned} v_5(N) &= \max(v_5(DT) * a_{DT \rightarrow N}; 0; 0) * b_N(WATCH) \\ &= (0,000108 * 0,9) * 0,1 = 9,72 * 10^{-6} \end{aligned} \quad \rightarrow \psi_5(N) = DT$$

$$\begin{aligned} v_5(V) &= \max(v_5(DT) * a_{DT \rightarrow V}; 0; 0) * b_V(WATCH) \\ &= (0,000108 * 0,1) * 0,3 = 3,24 * 10^{-6} \end{aligned} \quad \rightarrow \psi_5(V) = DT$$



Most likely state sequence:  
**DT → N → V → DT → N**  
 with probability:  
 **$9,72 * 10^{-6}$**

observations  
 $t \rightarrow$


states  
 $i \downarrow$

|    | V | THE  | FANS   | WATCH   | THE      | RACE                 |
|----|---|------|--------|---------|----------|----------------------|
| DT |   | 0,16 | 0      | 0       | 0,000108 | 0                    |
| N  |   | 0    | 0,0144 | 0,00216 | 0        | $9,72 \cdot 10^{-6}$ |
| V  |   | 0    | 0,0032 | 0,00108 | 0        | $3,24 \cdot 10^{-6}$ |



## 7.5.4 Evaluating POS taggers

Percent correct metric does not control for how easy the tagging task is. *Kappa* can be used instead when comparing a tagger to a standard or when comparing human labelers to each other.


 Kappa is the ratio of the proportion of times that two classifiers agree to the maximum proportion of times that the classifiers could agree. Both are *corrected for chance agreement*.

$$K = \frac{P(A) - P(E)}{1 - P(E)}$$

$P(A)$  is the proportion of times that the hypothesis agrees with the standard → *actual agreement*.  $P(E)$  is the expected agreement by chance (e.g.: for two labels: 50%).

For example, getting 60% of labels correct in a 50/50 decision would only get a score of:

$$K = \frac{0,6 - 0,5}{1 - 0,5} = \frac{0,1}{0,5} = 0,2$$

  $K > 0,8$  is often considered a good agreement.



## 8 ML Basics: Metrics for evaluation

### Confusion Matrix

Counts the correct and false classifications as basis for calculating different performance metrics.

$$accuracy = \frac{\text{correct predictions}}{\text{all predictions}} = \frac{TP+TN}{TP+TN+FP+FN}$$

$$errorRate = 1 - accuracy$$

Confusion Matrix

|              | PREDICTED CLASS |                    |                    |
|--------------|-----------------|--------------------|--------------------|
| ACTUAL CLASS |                 | Class=Yes          | Class=No           |
|              | Class=Yes       | True<br>Positives  | False<br>Negatives |
|              | Class=No        | False<br>Positives | True<br>Negatives  |

### Class imbalance problem

Sometimes classes have very unequal frequency, e.g., in fraud detection (98% ok vs. 2% fraud). The class of interest is commonly called the *positive class* and the rest *negative classes*.

### Precision $p$

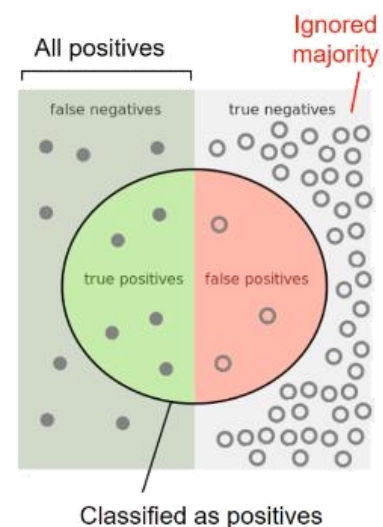
The number of *correctly classified positive* examples divided by the number of examples that are classified as positive.

$$p = \frac{TP}{TP + FP}$$

### Recall $r$

The number of *correctly classified positive* examples divided by the total number of actual positive examples in the test set.

$$r = \frac{TP}{TP + FN}$$



True Negatives get ignored as they are the majority that dilute the accuracy. These metrics (both equally important) should always be applied if there is a class imbalance problem.

### F<sub>1</sub>-Score

Combines *precision* and *recall* into one measure; it is the harmonic mean of both. It tends to be closer to the smaller of the two, so both must be large for a large F<sub>1</sub>-Score.

$$F_1 = \frac{2pr}{p+r} = \frac{2TP}{2TP + FP + FN}$$

! All measures above basically just count through the confusion matrix. This implies that every case has the same weight. So, errors, like false positives and false negatives are regarded as equally bad as well as the correct cases (true positives/negatives).

## 9 Vector Semantics

### 9.1 One-hot encoding

Represents words as one-hot vectors. The vector dimension is equal to the size of the vocabulary.

! There is no notion of word similarity/meaning.

**Motel:**  $[0, 0, 0, 0, 1, 0, 0, 0, 0, 0]$

**Hotel:**  $[0, 0, 0, 0, 0, 0, 0, 1, 0, 0]$

### 9.2 Distributional semantics

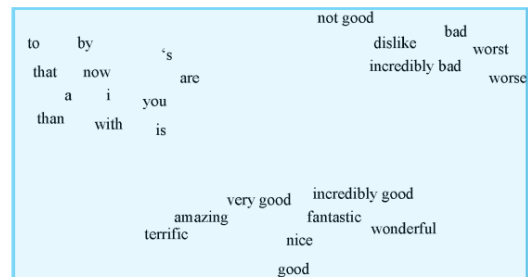
Words have similar meanings when they appear in the same context. Therefore, many contexts of a word are used to build up a representation for the word.

💡 Contexts can be sentences, paragraphs, documents or even metadata like geolocation.

By defining the meaning of a word as a vector they can be represented as a single point in a n-dimensional space. Similar words would be nearby in this vector space.

📄 Representing words in a vector space is a standard process in NLP, called *embedding*.

A bottle of *tesguino* is on the table  
Everybody likes *tesguino*  
*Tesguino* makes you drunk  
We make *tesguino* out of corn.



### 9.3 Cosine Similarity

📄 Tells how similar/different two vectors are by calculating the angle between two vectors. If the angle is  $0^\circ$  (the vectors have the exact same direction), the cos.-similarity is:  $\cos(0^\circ) = 1$ .

#### Formula

$$\text{cosine}(\vec{v}, \vec{w}) = \frac{\vec{v} \cdot \vec{w}}{|\vec{v}| \cdot |\vec{w}|} = \frac{\sum_{i=1}^N v_i \cdot w_i}{\sqrt{\sum_{i=1}^N v_i^2} \cdot \sqrt{\sum_{i=1}^N w_i^2}}$$

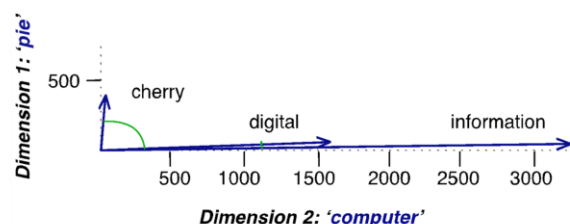
#### Data/Counts

|       |             | context words |      |          |
|-------|-------------|---------------|------|----------|
|       |             | pi            | data | computer |
| terms | cherry      | 442           | 8    | 2        |
|       | digital     | 5             | 1683 | 1670     |
|       | information | 5             | 3982 | 3325     |


#### Calculation

$$\begin{aligned} \text{cosine}(\vec{\text{cherry}}, \vec{\text{information}}) &= \frac{442 \cdot 5 + 8 \cdot 3982 + 2 \cdot 3325}{\sqrt{442^2 + 8^2 + 2^2} \cdot \sqrt{5^2 + 3982^2 + 3325^2}} = 0,17 \end{aligned}$$

$$\begin{aligned} \text{cosine}(\vec{\text{digital}}, \vec{\text{information}}) &= \frac{5 \cdot 5 + 1683 \cdot 3982 + 1670 \cdot 3325}{\sqrt{5^2 + 1683^2 + 1670^2} \cdot \sqrt{5^2 + 3982^2 + 3325^2}} = 0,996 \end{aligned}$$



## 9.4 Term frequency-inverse document frequency (TF-IDF)

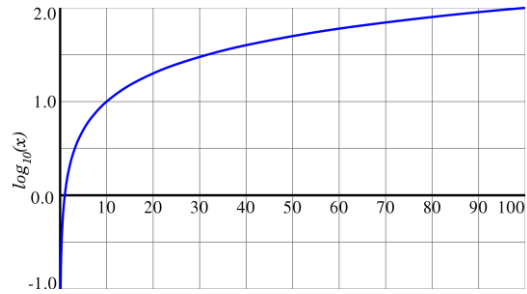
  $tf \times idf$  measures the importance of a term for a document, relative to the entire corpus.

*Term frequency* measures how frequently the word  $t$  appears in the document  $d$ .


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

The *inverse document frequency*  $idf$  decreases proportionally to the number of documents in a corpus that contain  $t$ .

$$idf = \log_{10}\left(\frac{N}{df}\right)$$



A high  $idf$  value therefore means that a word is very unique/specific as it does not appear in many documents. The word “the” would have a very low value as it appears very often.

 The  $\log_{10}()$ -function simply decreases the growth of the y-value for larger x-values.

### Example

Given the documents  $d_1, d_2, d_3, d_4$ :

- $d_1$ : I really enjoy having a **cold** **beer** at the **beach**.
- $d_2$ : **Ice** cream goes well with **beer**. In particular if it has **beer** flavor.
- $d_3$ : It would be great to sit at a **beach**. With an occasional, **cold**, refreshing breeze blowing through my hair, and an **ice** cream cone in my hand.
- $d_4$ : **Cold** **beer** is better than frozen yogurt is better than frozen **beer**.
- $d_5$ : Frozen water in cubic shape is called **ice** cube. Don't put **ice** cubes into **beer**. Or onto **ice** cream.
- $d_6$ : Yogurt **ice** cream is the best **ice** cream flavor.

1. Compute the term-document matrix for the words **cold**, **ice**, **beer**, **beach**.

$$idf_i = \log_{10}\left(\frac{N}{df_i}\right)$$

$\xrightarrow{\text{total \# of docs}}$   
 $\xrightarrow{\text{\# of docs with word } i}$

| term-doc.<br>counts | $d_1$ | $d_2$ | $d_3$ | $d_4$ | $d_5$ | $d_6$ | $df_i$ | $idf_i$                                     |
|---------------------|-------|-------|-------|-------|-------|-------|--------|---|
| <b>cold</b>         | 1     | 0     | 1     | 1     | 0     | 0     | 3      | $\log_{10}\left(\frac{6}{3}\right) = 0,301$ |
| <b>ice</b>          | 0     | 1     | 1     | 0     | 3     | 2     | 4      | $\log_{10}\left(\frac{6}{4}\right) = 0,176$ |
| <b>beer</b>         | 1     | 2     | 0     | 2     | 1     | 0     | 4      | $\log_{10}\left(\frac{6}{4}\right) = 0,176$ |
| <b>beach</b>        | 1     | 0     | 1     | 0     | 0     | 0     | 2      | $\log_{10}\left(\frac{6}{2}\right) = 0,477$ |

2. Compute the  $tf$ -matrix for the words *ice*, *beer*.


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

| $tf$        | $d_1$                  | $d_2$                      | $d_3$                  | $d_4$                      | $d_5$                      | $d_6$                      |
|-------------|------------------------|----------------------------|------------------------|----------------------------|----------------------------|----------------------------|
| <b>ice</b>  | 0                      | $1 + \log_{10}(1) = 1$     | $1 + \log_{10}(1) = 1$ | 0                          | $1 + \log_{10}(3) = 1,477$ | $1 + \log_{10}(2) = 1,301$ |
| <b>beer</b> | $1 + \log_{10}(1) = 1$ | $1 + \log_{10}(2) = 1,301$ | 0                      | $1 + \log_{10}(2) = 1,301$ | $1 + \log_{10}(1) = 1$     | 0                          |

3. Compute the  $tf \times idf$  matrix for the words *ice*, *beer*.


| $tf-idf$    | $d_1$ | $d_2$                   | $d_3$ | $d_4$                   | $d_5$                   | $d_6$                   |
|-------------|-------|-------------------------|-------|-------------------------|-------------------------|-------------------------|
| <b>ice</b>  | 0     | 0,176                   | 0,176 | 0                       | $0,176 * 1,477 = 0,256$ | $0,176 * 1,301 = 0,229$ |
| <b>beer</b> | 0,176 | $0,176 * 1,301 = 0,229$ | 0     | $0,176 * 1,301 = 0,229$ | 0,176                   | 0                       |


## 9.5 Pointwise mutual information (PMI)

 Alternative to  $tf-idf$ . It asks whether a context word is particularly informative about the target word. In other words, how much more often do words occur together compared to random occurrence.

- $P(x), P(y)$  is the likelihood of occurrence of word  $x, y$  overall
- $P(x, y)$  is the likelihood of  $x$  and  $y$  both occur in the same context

$$PMI(x, y) = \log_2 \frac{P(x, y)}{P(x) * P(y)}$$

 Problem: PMI ranges from  $(-\infty; \infty)$ , but negative values are problematic as things are co-occurring less than we expect by chance  $\rightarrow$  unreliable without enormous corpora

 Solution: Replace negatives values with zero  $\rightarrow$  positive PMI (PPMI):  $\max\left(\log_2 \left(\frac{P(x,y)}{P(x)*P(y)}\right); 0\right)$

Example (1/2)

| sample (1/2) |      | context words |      |        |           |        |    |
|--------------|------|---------------|------|--------|-----------|--------|----|
| terms        |      | document      | text | mining | analytics | lyrics | Σ  |
|              | song | 1             | 9    | 1      | 2         | 6      | 19 |
|              | text | 6             | 3    | 6      | 8         | 2      | 25 |
|              | web  | 3             | 5    | 6      | 2         | 1      | 17 |
|              | word | 5             | 2    | 3      | 1         | 3      | 14 |
|              | pen  | 2             | 2    | 0      | 0         | 1      | 5  |
|              | Σ    | 17            | 21   | 16     | 13        | 13     | 80 |

$PPMI(\text{text}, \text{analytics})$

$$= \max\left(\log_2 \left(\frac{P(\text{text}, \text{analytics})}{P(\text{text}) * P(\text{analytics})}\right); 0\right) = \max\left(\log_2 \left(\frac{\frac{8}{80}}{\frac{25}{80} * \frac{13}{80}}\right); 0\right) = 0,978$$

❗ Problem: PMI is biased toward infrequent events; very rare words have higher probabilities.

💡 Solution: Use add-one smoothing or give rare words slightly higher probabilities.

### Example (2/2)

|       |          | context words |        |           |        |          |        |
|-------|----------|---------------|--------|-----------|--------|----------|--------|
| terms | document | text          | mining | analytics | lyrics | $\Sigma$ |        |
|       | song     | 1 +1          | 9 +1   | 1 +1      | 2 +1   | 6 +1     | 19 +5  |
|       | text     | 6 +1          | 3 +1   | 6 +1      | 8 +1   | 2 +1     | 25 +5  |
|       | web      | 3 +1          | 5 +1   | 6 +1      | 2 +1   | 1 +1     | 17 +5  |
|       | word     | 5 +1          | 2 +1   | 3 +1      | 1 +1   | 3 +1     | 14 +5  |
|       | pen      | 2 +1          | 2 +1   | 0 +1      | 0 +1   | 1 +1     | 5 +5   |
|       | $\Sigma$ | 17 +5         | 21 +5  | 16 +5     | 13 +5  | 13 +5    | 80 +25 |

$$PPMI_{add1}(text, analytics)$$

$$= \max \left( \log_2 \left( \frac{P(song, text)}{P(song) * P(text)} \right); 0 \right) = \max \left( \log_2 \left( \frac{\frac{10}{105}}{\frac{24}{105} * \frac{26}{105}} \right); 0 \right) = 0,751$$

❓ PMI representations are long ( $|V|$  ranges from 20.000 to 50.000) and sparse (most entries are 0). An alternative are dense vectors, which are shorter and contain less zeros. They may be easier to use as features and may generalize better.

## 10 Foundations of Neural Networks

- See GNN & Data Mining I script

# 11 Word Embeddings

## 11.1 Word2Vec

A framework for learning word vectors. Instead of counting how often each word  $w$  occurs near a particular word  $x$  a classifier is trained on a binary prediction task. Is  $w$  likely to show up near  $x$ ? The learned classifier weights will be used as the word embeddings.

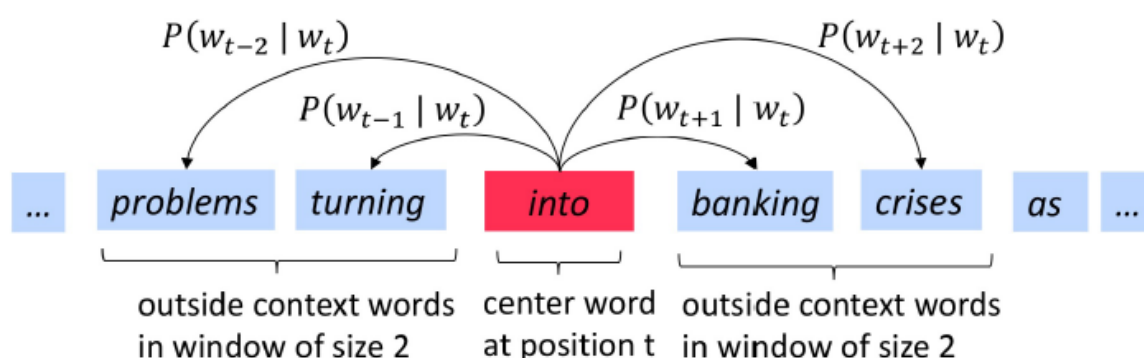
Given a large corpus of text, every word in a fixed vocabulary is represented by a vector. Use the similarity of the word vectors for  $w_t$  (center word) and  $w_{t \pm j}$  (context of size  $t$ ) to calculate the probability of  $w_{t \pm j}$  given  $w_t$ .

CBOW (Continuous Bag of Words)

- Predict the **center word** from the **context words**

Skip-gram

- Predict the **context words** from the **center word**



### 11.1.1 Matrix multiplication and dot product

To define multiplication between a matrix  $A$  and a vector  $\vec{v}$  (i.e., the matrix-vector product), we need to view the vector as a column matrix. We define the matrix-vector product only for the case when the number of columns in  $A$  equals the number of rows in  $\vec{v}$ . So, if  $A$  is an  $m \times n$  matrix (i.e., with  $n$  columns), then the product  $A * \vec{v}$  is defined for  $n \times 1$  column vectors  $\vec{v}$ .

If we let  $A * \vec{v} = b$ , then  $b$  is an  $m \times 1$  column vector. In other words, the number of rows in  $A$  (which can be anything) determines the number of rows in the product  $b$ .

$$A * \vec{v} = \vec{v} * A = (x, y, z) * \begin{pmatrix} 1 & 2 & 3 \\ 4 & -5 & 6 \\ -7 & 8 & 9 \end{pmatrix} = \begin{pmatrix} x + 2y + 3z \\ 4x - 5y + 6z \\ -7x + 8y + 9z \end{pmatrix}$$

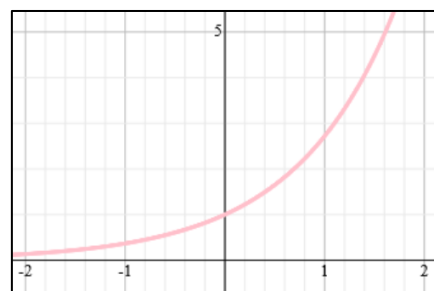
Although it may look confusing at first, the process of matrix-vector multiplication is actually quite simple. One takes the dot product of  $\vec{v}$  with each of the rows of  $A$ . (This is why the number of columns in  $A$  has to equal the number of components in  $\vec{v}$ .) The first component of the matrix-vector product is the dot product of  $\vec{v}$  with the first row of  $A$ , etc. In fact, if  $A$  has only one row, the matrix-vector product is really a dot product in disguise.

### 11.1.2 SoftMax function

The SoftMax is a very frequent function in Deep Learning. It maps arbitrary values  $x_i$  to a probability distribution  $p_i$ , therefore transforming the output of a neural network into class probabilities. Generally, it has this form:

$$\text{softmax}(x_i) = \frac{e^{x_i}}{\sum_{j=1}^N e^{x_j}}$$

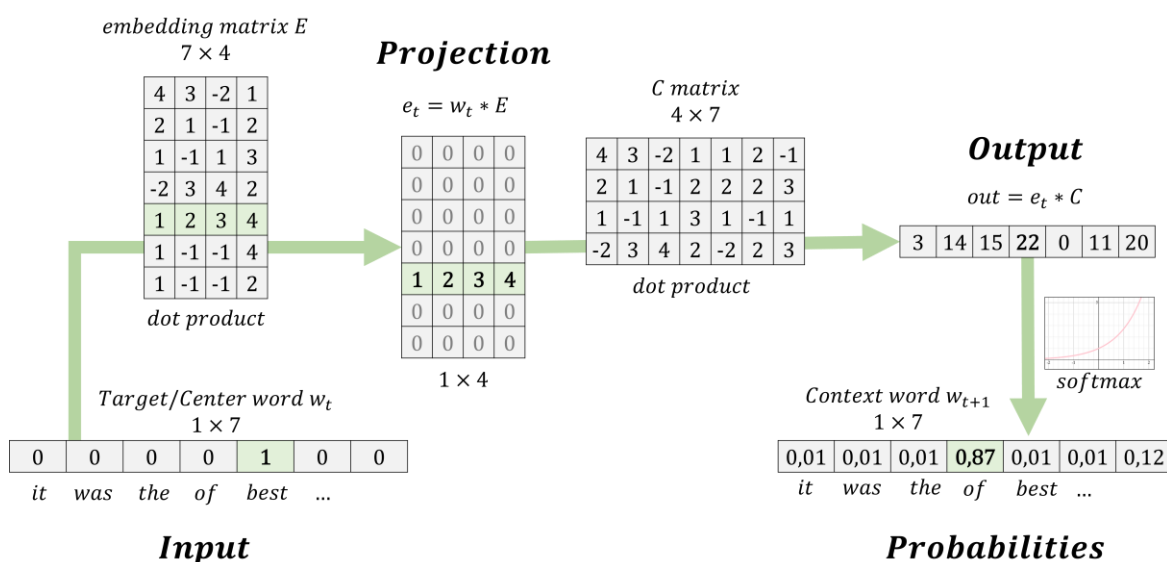
The e-function preserves order and turns negative value into positives. The denominator sums all the values to ensure that the final result of the SoftMax-function will sum to 1.



$e^x$ -function

### 11.1.3 Neural Network

Once trained, the embedding  $E$  provides the representations for the words in the vocabulary.



To train the network we have to translate it to an equation with parameters to optimize:

$$P(w_{t+j}|w_t) = \frac{\exp(e_t * c_{t+j}^T)}{\sum_{w \in V} \exp(e_t * c_w^T)}$$

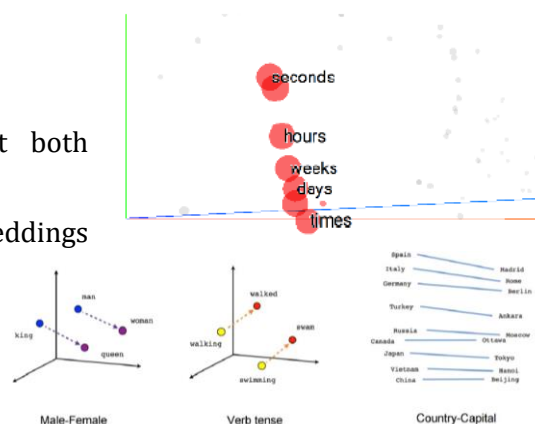
To optimize the weights, we define a quality function for the current network weights  $\theta$  and optimize this with gradient descent.

### 11.1.4 Findings


Embeddings created by Word2Vec represent both semantic similarity and semantic relations.

Similarity means, that words with similar embeddings have similar meaning. So, words with similar embeddings have similar contexts.

Relations are often encoded as a direction in the embedding.




## 11.2 Global Vectors (GloVe)

 GloVe is another method for creating word embeddings from a corpus. It is calculated over the global word co-occurrence matrix and the resulting embeddings are of similar quality to Word2Vec.

|   | $x = \text{solid}$ | $x = \text{gas}$ | $x = \text{water}$ | $x = \text{random}$ |
|---|--------------------|------------------|--------------------|---------------------|
| $P(x \text{ice})$                           | large              | small            | large              | small               |
| $P(x \text{steam})$                         | small              | large            | large              | small               |
| $\frac{P(x \text{ice})}{P(x \text{steam})}$ | large              | small            | $\sim 1$           | $\sim 1$            |

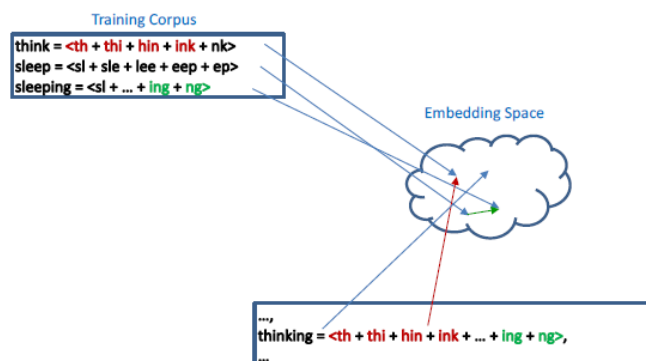
 A crucial insight is that ratios of co-occurrence probabilities can encode meaning.

## 11.3 Fasttext

 Fasttext is a library for word embeddings considering sub-word information.

This helps both Word2Vec and GloVe as previously unseen records cannot be embedded.

Its intuition is, that unknown words can be modeled by their n-grams.



## 11.4 Evaluation

 How can two word embeddings be compared?

### 11.4.1 Intrinsic

- Evaluation on a specific/ intermediate task
- Fast to compute
- Helps to understand the system

### 11.4.2 Extrinsic

- Evaluation on a real task
- Can take a long time to compute accuracy