Language modeling

IN4080
Natural Language Processing

Yves Scherrer

Save the date

Presentation of master thesis topics

- Monday October 9th at 15:00
- Seminar room Perl

Mandatory assignment 2

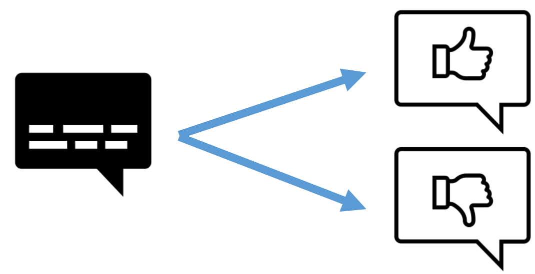
- Will be published by tomorrow
- Submission deadline: Monday October 16th

Main NLP tasks

Natural language generation

- Machine translation
- Question answering
- Grammatical error correction
- …

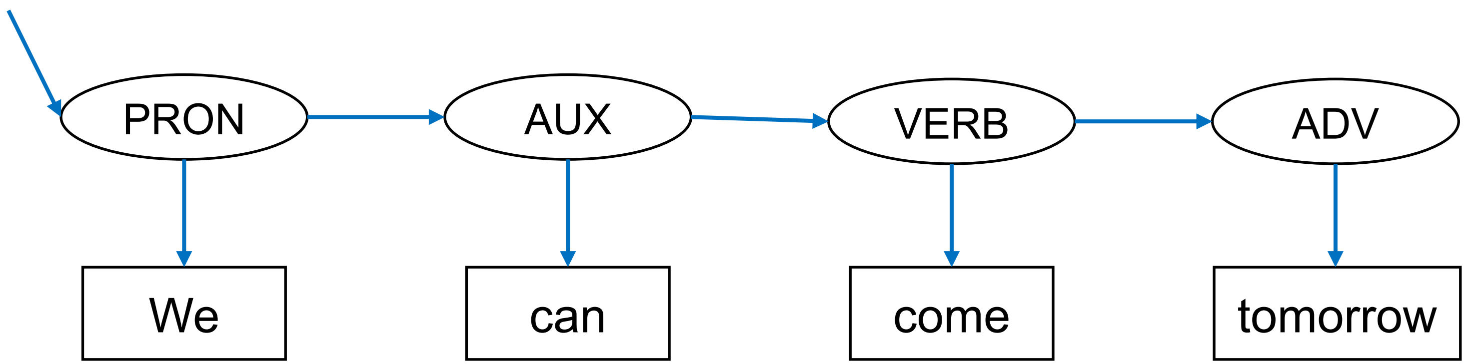
Annotation (natural language understanding)



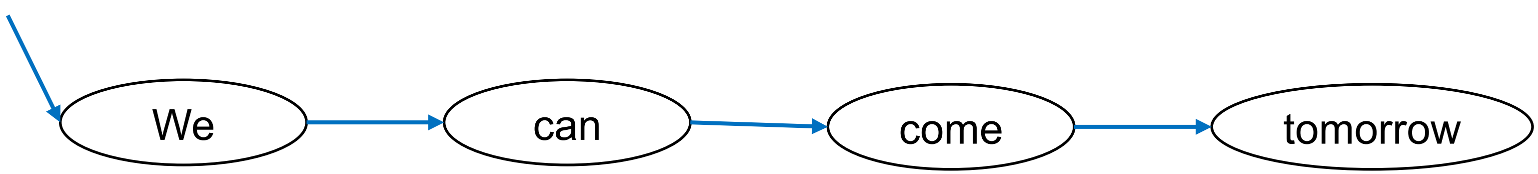
- Hate speech detection
- Sentiment analysis
- Language identification
- Syntactic analysis
- …

Sequence labeling and language modeling

HMM for sequence labeling:



Bigram language model:



$P(\text{We}|*)$ $P(\text{can}|\text{We})$ $P(\text{come}|\text{can})$ $P(\text{tomorrow}|\text{come})$

What can we do with language models?

- Assign probabilities to sentences
  - Choose among different hypotheses
    - Disambiguation, reranking
  - Score the same sentence with different language models
    - Language identification, typology
- Predict the probability of the next word
    - Text completion, spelling correction
- Generate entirely new sentences
    - Mostly for fun ☺

Probabilistic language models

- Assign a probability to the sentence $w_{1..n}$:
$$P(w_1, w_2, w_3, \dots, w_n)$$
  - We can only do that reliably if we have seen this exact sentence (several times) in the training data
  - This is unlikely for most of the sentences
- Chain rule:
$$P(w_1, w_2, w_3, \dots, w_n)$$
$$= P(w_1) \cdot P(w_2 | w_1) \cdot P(w_3 | w_1, w_2) \cdot \cdots$$
$$\cdot P(w_n | w_1, w_2, \dots, w_{n-1})$$
- (Bigram) Markov assumption:
$$P(w_i | w_1, \dots, w_{i-1}) \approx P(w_i | w_{i-1})$$

Probabilistic language models

- Bigram language model:

$$P(w_1, w_2, w_3, \dots, w_n)$$
$$\approx P(w_1 | *) \cdot P(w_2 | w_1) \cdot \dots \cdot P(w_n | w_{n-1})$$

Sequences of 2 words

- Trigram language model:

$$P(w_1, w_2, w_3, \dots, w_n)$$
$$\approx P(w_1 | *,*) \cdot P(w_2 | *, w_1) \cdot \dots \cdot P(w_n | w_{n-2}, w_{n-1})$$

Sequences of 3 words

- Four-gram language model:

Sequences of 4 words

$$P(w_1, w_2, w_3, \dots, w_n)$$
$$\approx P(w_1 | *,*,*) \cdot P(w_2 | *,*, w_1) \cdot \dots \cdot P(w_n | w_{n-3}, w_{n-2}, w_{n-1})$$

Estimating the probabilities

- Maximum likely estimates for a bigram LM:

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

- We can add some smoothing:

$$\hat{P}(w_i|w_{i-1}) = \frac{Count(w_{i-1}, w_i) + \alpha}{Count(w_{i-1}) + \alpha \cdot |V|}$$

- Note: we assume here that the w_i are words. One can also use individual characters.

Example

- 3 sentences, with start and end symbols:
  - Bigram probabilities, no smoothing

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

- $P(\mathrm{I} | \langle s \rangle) =$
- $P(\mathrm{Sam} | \mathrm{am}) =$
- $P(\mathrm{do} | \mathrm{I}) =$
- $P(\langle /s \rangle | \mathrm{Sam}) =$
- $P(\mathrm{eggs} | \mathrm{ham}) =$

More smoothing techniques

Additive smoothing provides non-zero probabilities for unknown n-grams.

- In many cases, the words constituting these n-grams are actually known.

Example:

- Shakespeare produced 884,647 word tokens and 29,066 word types (V)
- This gives a theoretical number of 844,000,000 possible bigram types (V^2)
- In Shakespeare’s work, only 300,000 bigram types are realized (0.035%)

More smoothing techniques

Backoff smoothing:

- Train several models of different orders on the same data and combine them.
- Example:
  - If you have good evidence, use the 4-gram model score,
  - If not, use the trigram model score,
  - If not, use the bigram model score,
  - If not, use the unigram model score:

$$P(w_1, w_2, w_3, \dots, w_n) \approx P(w_1) \cdot P(w_2) \cdot \cdots \cdot P(w_n)$$

More smoothing techniques

Interpolation:

- Always use a combination of all models with different weights:

$$
\begin{aligned}
&P(w_i|w_{i-3}, w_{i-2}, w_{i-1}) \\
&= \lambda_4 \cdot P_4(w_i|w_{i-3}, w_{i-2}, w_{i-1}) \\
&+ \lambda_3 \cdot P_3(w_i|w_{i-2}, w_{i-1}) \\
&+ \lambda_2 \cdot P_2(w_i|w_{i-1}) \\
&+ \lambda_1 \cdot P_1(w_i)
\end{aligned}
$$

Four-gram model

Trigram model

Bigram model

Unigram model

- Note: $\lambda_4 + \lambda_3 + \lambda_2 + \lambda_1 = 1$

More smoothing techniques

Kneser-Ney smoothing:

- See J&M 3.7

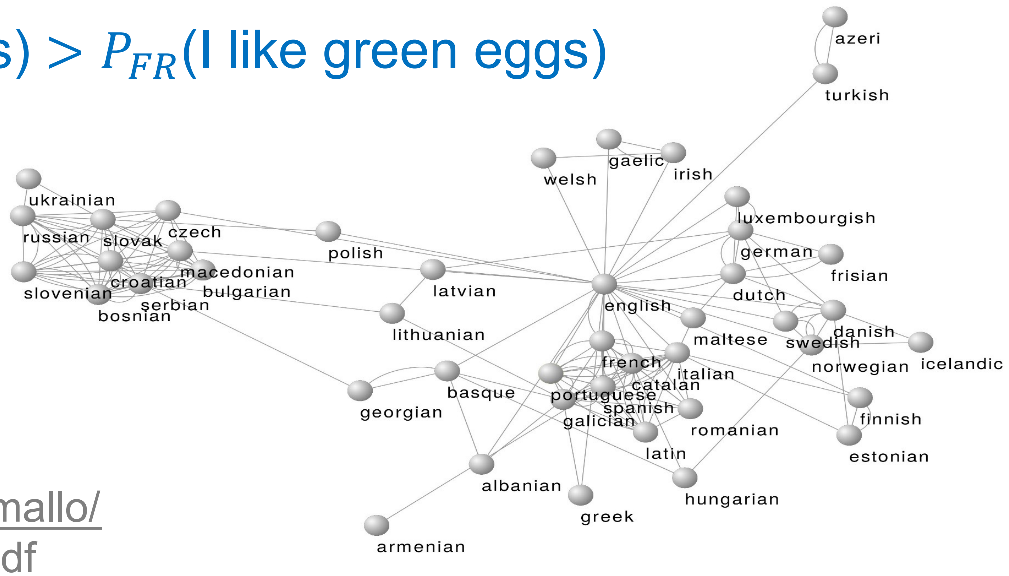
What can we do with language models?

- Assign probabilities to sentences
  - Choose among different hypotheses

  - Translation:
    - P(she is a tall woman) > P(she is a high woman)
    - P(she has a high position) > P(she has a tall position)
  - Spelling correction:
    - P(She met the prefect.) > P(She met the perfect.)
    - P(She met the prefect match.) < P(She met the perfect match.)
  - Speech recognition:
    - P(I saw a van) > P(eyes awe of an)

What can we do with language models?

- Assign probabilities to sentences
  - Score the same sentence with different language models
  - Language identification:
    - P_{EN}(I like green eggs) $>$ P_{FR}(I like green eggs)
  - Compute distances between languages



https://gramatica.usc.es/~gamallo/artigos-web/PHYSICA2017.pdf

What can we do with language models?

- Predict the probability of the next word
  - Predictive text on phones

https://support.apple.com/en-in/guide/iphone/iphd4ea90231/ios

What can we do with language models?

- Generate entirely new sentences
  - Sample w_1 according to $\hat{P}(w_1|\langle s\rangle)$
  - Sample w_2 according to $\hat{P}(w_2|w_1)$
  - …

Models trained on Shakespeare texts (J&M Fig. 3.4)

1 gram	–To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have –Hill he late speaks; or! a more to leg less first you enter
2 gram	–Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow. –What means, sir. I confess she? then all sorts, he is trim, captain.
3 gram	–Fly, and will rid me these news of price. Therefore the sadness of parting, as they say, ’tis done. –This shall forbid it should be branded, if renown made it empty.
4 gram	–King Henry. What! I will go seek the traitor Gloucester. Exeunt some of the watch. A great banquet serv’d in; –It cannot be but so.

What can we do with language models?

- Generate entirely new sentences
  - Character-level n-gram language models:

Bigram character LM:
Fif thad yourty
Fare sid on Che as al my he sheace ing.
Thy your thy ove dievest sord wit whand of sold iset?
Commet laund hant.
KINCESARGANT:
Out aboy tur Pome you musicell losts, blover.
How difte quainge to sh,
And usbas ey will Chor bacterea, and mens grou:

Four-gram character LM:
First Office, masters
To part at that she may direct my brance
I would he dead. Pleaseth profit,
Then we last awaked you to again,
Far that night I'll courteous Herneath,
Of circle off.
SPEED:
Not you.

Ten-gram character LM:
First Citizen:
Nay, then, that was hers,
It speaks against your other service:
But since the
youth of the circumstance be spoken:
Your uncle and one Baptista's daughter.
SEBASTIAN:
Do I stand till the break off.
BIRON:
Hide thy head.

https://nbviewer.org/gist/yoavg/d76121dfde2618422139

Evaluation of language models

Extrinsic evaluation:

- To compare two LMs A and B, see how well they are doing in an application
  - Machine translation
  - Speech recognition
- Run the application with A and B, get accuracy figures, determine which one does better

Evaluation of language models

Intrinsic evaluation:

- Use a held-out corpus and measure the probabilities given to it by A and B.
- The best language model is the one that best predicts the unseen corpus.
- Probability: $P(w_1, w_2, \dots, w_n)^{\frac{1}{n}}$
  - The n-root compensates for different sentence lengths
- **Perplexity**: the inverse probability of the test set, normalized by the number of words:
  $PP(w_1, w_2, \dots, w_n) = P(w_1, w_2, \dots, w_n)^{-\frac{1}{n}}$

Perplexity

- A better model of a text is one which assigns higher probabilities to the words that actually occur in the text.
- Perplexity gives an “average” over all words in a sentence.
  - Minimize perplexity = maximize probability
- Perplexity can be used as a language distance measure:
  - Perplexity of applying an English language model on French ≈ linguistic distance between English and French

What can we do with n-gram language models?

- ☑ Assign probabilities to sentences
- ☑ Predict the probability of the next word
- ☒ Generate entirely new sentences
  - N-gram models are just too bad for generating useful text:
    - No long-distance dependencies
    - No explicit syntax, no hierarchical structure
  - Neural LMs can do much better!

Word vectors and embeddings

IN4080

Natural Language Processing

Yves Scherrer

The meaning of words

lemma

sense

definition

pepper, *n.*

Pronunciation: Brit. /ˈpɛpə/ , U.S. /ˈpɛpər/

Forms: OE **peopor** (*rare*), OE **pipcer** (transmission error), OE **pipor**, OE **pipur** (*rare* .

Frequency (in current use):

Etymology: A borrowing from Latin. **Etymon:** Latin *piper*.
< classical Latin *piper*, a loanword < Indo-Aryan (as is ancient Greek *πέπερι*); compare Sar

I. The spice or the plant.

1.

a. A hot pungent spice derived from the prepared fruits (peppercorns) of the pepper plant, *Piper nigrum* (see sense 2a), used from early times to season food, either whole or ground to powder (often in association with salt). Also (locally, chiefly with distinguishing word): a similar spice derived from the fruits of certain other species of the genus *Piper*; the fruits themselves.

The ground spice from *Piper nigrum* comes in two forms, the more pungent *black pepper*, produced from black peppercorns, and the milder *white pepper*, produced from white peppercorns: see BLACK *adj.* and *n.* Special uses 5a, PEPPERCORN *n.* 1a, and WHITE *adj.* and *n.*[1] Special uses 7b(a).

2.

a. The plant *Piper nigrum* (family Piperaceae), a climbing shrub indigenous to South Asia and also cultivated elsewhere in the tropics, which has alternate stalked entire leaves, with pendulous spikes of small green flowers opposite the leaves, succeeded by small berries turning red when ripe. Also more widely: any plant of the genus *Piper* or the family Piperaceae.

b. Usu. with distinguishing word: any of numerous plants of other families having hot pungent fruits or leaves which resemble pepper (1a) in taste and in some cases are used as a substitute for it.

c. *U.S.* The California pepper tree, *Schinus molle*. Cf. PEPPER TREE *n.* 3.

3. Any of various forms of capsicum, esp. *Capsicum annuum* var. *annuum*. Originally (chiefly with distinguishing word): any variety of the *C. annuum* Longum group, with elongated fruits having a hot, pungent taste, the source of cayenne, chilli powder, paprika, etc., or of the

- A word with several senses is called **polysemous**.
- If two different words look and sound the same, they are called **homonyms**.

- How to tell the difference between one word with several senses, and two words?
- Etymology (common origin) may help, but not always…

Relations between senses

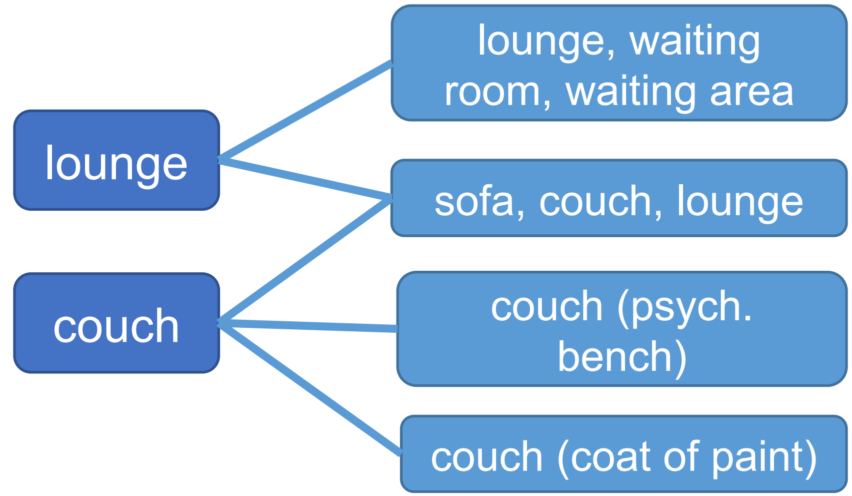
The different senses of a word can be related in various ways:

- **Synonyms**: have the same meaning in all (or at least some) contexts
  - sofa – couch, bus – coach, big – large
- **Antonyms**: opposites with respect to a feature of meaning
  - true – false, strong – weak, up – down
- **Hyponyms and hypernyms**: the <*hyponym*> is a type of the <*hypernym*>
  - rose → flower, cow → animal, car → vehicle

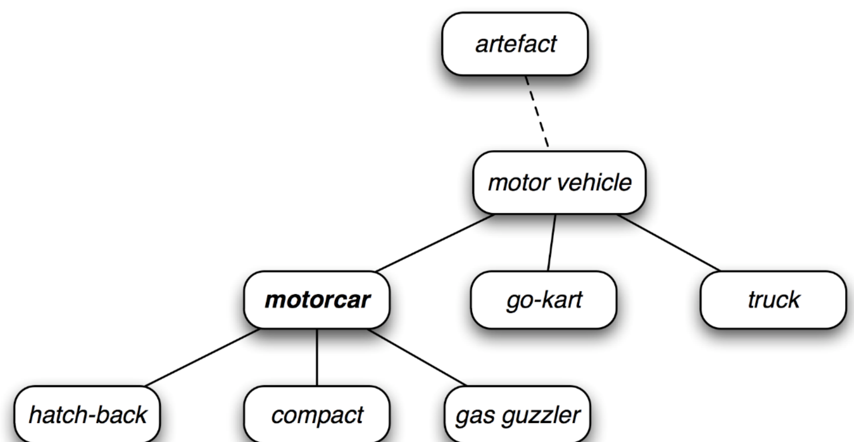
Resources for lexical semantics

WordNet: https://wordnet.princeton.edu

- Words are grouped into synsets



- Hyponymy relations between the synsets



Relations between senses

Less well defined relations between senses:

- **Similarity**: have a common hypernym
  - cow – horse, boy – girl
- **Relatedness**
  - money – bank, fish – water

What does ong choi mean?

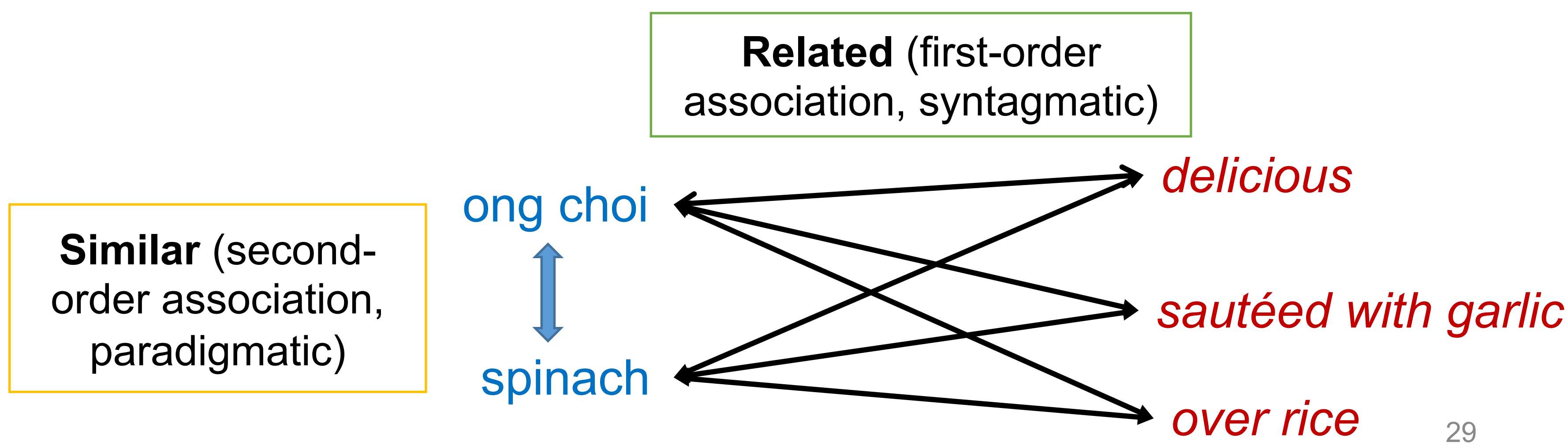
- Suppose you see these sentences:
  - *Ong choi is delicious sautéed with garlic.*
  - *Ong choi is superb over rice*
  - *Ong choi leaves with salty sauces*
- And you've also seen these:
  - *…spinach sautéed with garlic over rice*
  - *Chard stems and leaves are delicious*
  - *Collard greens and other salty leafy greens*
- Conclusion: Ongchoi is a leafy green like spinach, chard, or collard greens

(J&M chapter 3)

Relations between senses

Less well defined relations between senses:

- **Similarity**: have a common hypernym
  - cow – horse, boy – girl
- **Relatedness**
  - money – bank, fish – water



Distributional word representations

The distributional hypothesis:

- The meaning of a word can be captured by the contexts in which it occurs.
- Words that occur in similar contexts have similar meanings.

Example (Nida 1975):

A bottle of tesgüino is on the table.
Tesgüino makes you drunk.
Everybody likes tesgüino.
We make tesgüino out of corn.

J.P. Firth, 1957:

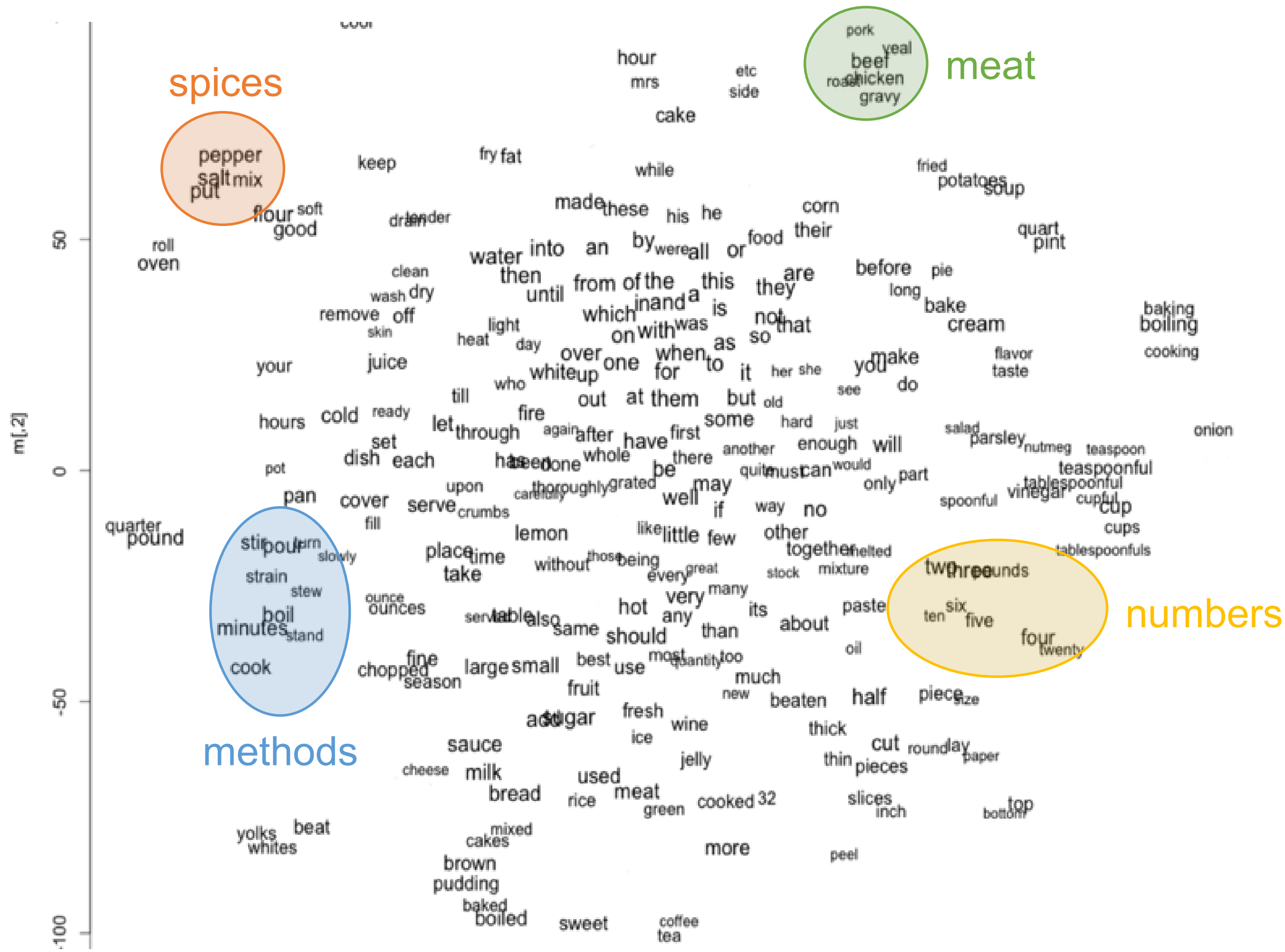
"You shall know a word by the company it keeps."

Vector semantics

Core idea:

- Each word is represented as a point in a (multidimensional) semantic space.
    - **Word vectors, word embeddings**
- The points are inferred from the distributions of word neighbors/contexts in text, according to the distributional hypothesis.
- Similar/related words are close to each other in this space.

Vector semantics



https://www.adityathakker.com

Vector semantics

How do we get there?

- Word-document (or term-document) matrices based on co-occurrence counts
  - Count weighting with tf-idf
- Word-context matrices based on co-occurrence counts
  - Dimensionality reduction (SVD, LSA)
- An alternative approach: word2vec

Term-document matrices

- One row per term/word
- One column per document
- Values represent counts of terms in documents

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

Figure 6.2 The term-document matrix for four words in four Shakespeare plays. Each cell contains the number of times the (row) word occurs in the (column) document.

The column vectors correspond to the bag-of-words representations for document classification.

Term-document matrices

What can we do with such a matrix?

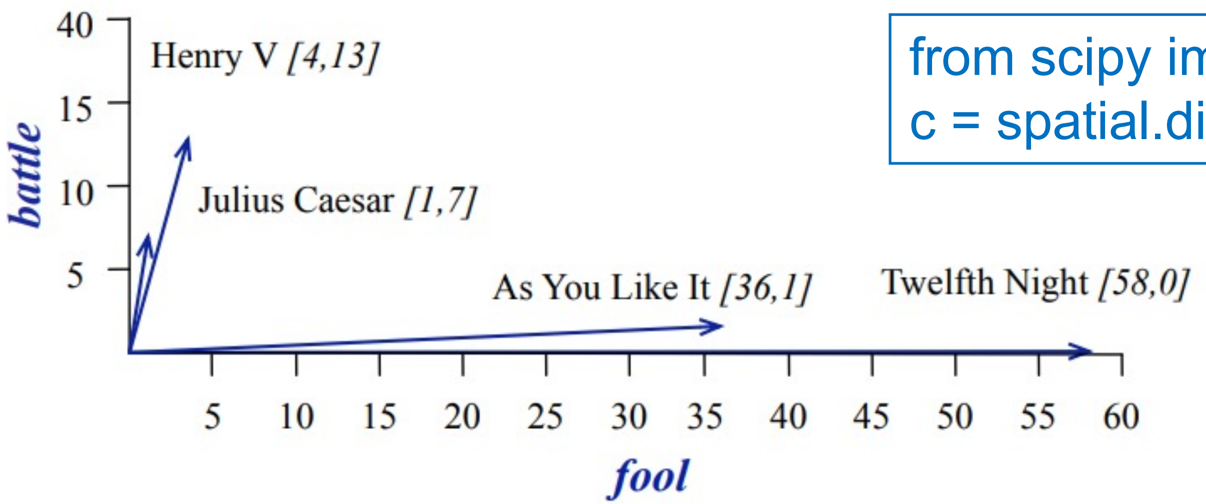
	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

- Compute similarity between words
  - *fool* and *wit* are similar
- Compute similarity between documents
  - *As You Like It* and *Twelfth Night* are similar (comedies)
  - *Julius Caesar* and *Henry V* are similar (historical dramas)

Cosine similarity

Cosine similarity represents the **angle** between two vectors:

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




```
from scipy import spatial
c = spatial.distance.cosine(v, w)
```

Term-document matrices

What can we do with such a matrix?

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

- Compute similarity between words
  - *fool* and *wit* are similar
  - $cosine(\text{fool}, \text{wit}) = cosine([36,58,1,4], [20,15,2,3]) = 0.93$
  - $cosine(\text{fool}, \text{battle}) = cosine([36,58,1,4], [1,0,7,13]) = 0.09$
- Compute similarity between documents
  - As You Like It and Twelfth Night are similar (comedies)
  - Julius Caesar and Henry V are similar (historical dramas)
  - $cosine(\text{AYLI}, \text{TN}) = cosine([1,114,36,20], [0,80,58,15]) = 0.95$
  - $cosine(\text{JC}, \text{HV}) = cosine([7,62,1,2], [13,89,4,3]) = 0.69$
  - $cosine(\text{TN}, \text{JC}) = cosine([0,80,58,15], [7,62,1,2]) = 0.81$

Term-document matrices

What can we do with such a matrix?

	As You Like It	Twelfth Night	Julius Caesar	Henry V	Q: good fool
battle	1	0	7	13	0
good	114	80	62	89	1
fool	36	58	1	4	1
wit	20	15	2	3	0

- Compute similarity between words
- Compute similarity between documents
- **Information retrieval:**
  - Encode the **query** as an additional document
  - Find documents that are most similar to the query

Term-document matrices

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

Each word is associated with a vector that describes its meaning:

- *battle* is "the kind of word that occurs in Julius Caesar and Henry V"
- *fool* is "the kind of word that occurs in comedies, especially Twelfth Night

Vector semantics

How do we get there?

- Word-document (or term-document) matrices based on co-occurrence counts
  - Count weighting with tf-idf
- Word-context matrices based on co-occurrence counts
  - Dimensionality reduction (SVD, LSA)
- An alternative approach: word2vec

Count weighting

Are all words equally important?

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

- Intuition: A word occurring in a large proportion of documents is not a good discriminator
  - *good* does not contribute much to distinguishing the documents
  - The importance of a word should be inversely proportional to the number of documents it occurs in

TF-IDF count weighting

- TF – term frequency:
  - $tf_{t,d}$ is the frequency of term t in document d
- DF – document frequency:
  - df_t is the number of documents containing term t
- IDF – inverse document frequency:
  - $idf_t = \frac{1}{df_t}$
  - Normalize by the collection size: $\frac{N}{df_t}$
  - By convention, take the log: $\log \frac{N}{df_t}$
- TF-IDF: $tf_{t,d} \cdot \log \frac{N}{df_t}$

Implementation: replace Scikit-Learn *CountVectorizer* by *TfIdfVectorizer*

TF-IDF count weighting

Raw counts:

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

TF-IDF weights:

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	0.074	0	0.22	0.28
good	0	0	0	0
fool	0.019	0.021	0.0036	0.0083
wit	0.049	0.044	0.018	0.022

TF-IDF did its job!

Vector semantics

How do we get there?

- Word-document (or term-document) matrices based on co-occurrence counts
  - Count weighting with tf-idf
- Word-context matrices based on co-occurrence counts
  - Dimensionality reduction (SVD, LSA)
- An alternative approach: word2vec

Word-context matrices

- One row per term/word
- One column per context term/word
  - The rows and columns may be the same, but do not have to
- Values represent counts of words **within the context** of another word

	aardvark	...	computer	data	result	pie	sugar	...
cherry	0	...	2	8	9	442	25	...
strawberry	0	...	0	0	1	60	19	...
digital	0	...	1670	1683	85	5	4	...
information	0	...	3325	3982	378	5	13	...

Word-context matrices

What is the context of a word?

- The same sentence
  - Add a pinch of sugar to the cherries and boil for 10 minutes.
  - In the digital age, information is the global currency.
- n words to the left and to the right
  - Add a pinch of [sugar to the cherries and boil for] 10 minutes.
  - In [the digital age, information is the global] currency.

$n = 3$

- Often, stopwords are not counted
  - Add a [pinch of sugar to the cherries and boil for 10] minutes.
  - In the [digital age, information is the global currency.]

$n = 2$

Word-context matrices

What can we do with such a matrix?

	aardvark	...	computer	data	result	pie	sugar	...
cherry	0	...	2	8	9	442	25	...
strawberry	0	...	0	0	1	60	19	...
digital	0	...	1670	1683	85	5	4	...
information	0	...	3325	3982	378	5	13	...

- Compute cosine similarity between words
  - *Cherry* and *strawberry* are similar
  - *Digital* and *information* are similar
- Compute cosine similarity between context words (columns)
  - Similar outcome as for rows, but not usually done

Vector semantics

How do we get there?

- Word-document (or term-document) matrices based on co-occurrence counts
  - Count weighting with tf-idf
- Word-context matrices based on co-occurrence counts
  - Dimensionality reduction (SVD, LSA)
- An alternative approach: word2vec

Dimensionality reduction

- Word-context vectors are sparse
  - Most values are 0
  - Most words never co-occur with *strawberry*
  - In particular when choosing small context sizes

	aardvark	...	computer	data	result	pie	sugar	...
cherry	0	...	2	8	9	442	25	...
strawberry	0	...	0	0	1	60	19	...
digital	0	...	1670	1683	85	5	4	...
information	0	...	3325	3982	378	5	13	...

- Drawbacks:
  - Inefficient
  - Lack generalization capabilities

Dimensionality reduction

- Reduce the number of columns to a fixed size m (typically in the range $50 \dots 500$)
  - e.g. using Singular Value Decomposition (SVD)
- The new columns represent abstract properties, not words:

	leash	walk	run	owner	leg	bark
dog	3	5	1	5	4	2
cat	0	3	3	1	5	0
lion	0	3	2	0	1	0
light	0	0	0	0	0	0
bark	1	0	0	2	1	0
car	0	0	4	3	0	0

	u_A	u_B	u_C	u_D
dog	0.2	0.3	0	0.7
cat	0.7	0.2	0	0.8
lion	0.9	0.1	0	0.8
light	0	0	0.7	0
bark	0.3	0.9	0.1	0.6
car	0	0.3	0.6	0

Dimensionality reduction

- The combination of word-context vectors with SVD dimensionality reduction is known as LSA (latent semantic analysis – Deerwester et al. 1990).
- Other combinations are possible, e.g. using PCA (principal component analysis).
- We typically use 100-500 target dimensions, but can use as few as 2 for visualization.

Word embeddings

- There is a direct way to obtain dense word-context vectors, by using neural networks: **word2vec**
  - Details a bit later in this course
- More recent developments:
  - **FastText**
  - Contextualized embeddings: **BERT**

- The Python module *gensim* is really useful for all kinds of word-vector-related experiments!

What can we do with word and document vectors?

Term-document vectors:

- Use document vectors as bag-of-words representation for text classification
  - TF-IDF weighting may be helpful
  - Dimensionality reduction can be applied
- Compute similarities between documents + find most similar documents to a given query
  - TF-IDF weighting is considered standard
  - Recent approaches use dense vectors from NNs

What can we do with word and document vectors?

Word-context vectors:

- Use word vectors as feature vectors for sequence labeling tasks
  - Dimensionality reduction is required, otherwise it is just a one-hot vector
- Compute similarities between words
  - Dimensionality reduction considered standard
  - Specific research questions: analogy, semantic change, bias detection

What can we do with word and document vectors?

Word-context vectors:

- Can we use word vectors as bag-of-words representation for text classification?
  - **Continuous bag-of-words model** (CBOW): Average the weight vectors of all words occurring in the document

$$\boldsymbol{v}_{\boldsymbol{avg}}(x) = \frac{1}{|x|} \cdot \sum_{w \in x} \boldsymbol{v}_{\boldsymbol{w}}$$

Readings

- Jurafsky & Martin, chapter 3
  - N-gram language models

- Jurafsky & Martin, chapter 6
  - Vector semantics and embeddings