UNIVERSITÄT
BAYREUTH

Modeling
Text and Language

Mirco Schönfeld
University of Bayreuth

mirco.schoenfeld@uni-bayreuth.de
@TWlyY29

Talk to ELIZA

User: I am unhappy.

ELIZA: DO YOU THINK COMING HERE WILL HELP YOU NOT TO BE UNHAPPY

User: I need some help, that much seems certain.

ELIZA: WHAT WOULD IT MEAN TO YOU IF YOU GOT SOME HELP

User: Perhaps I could learn to get along with my mother.

ELIZA: TELL ME MORE ABOUT YOUR FAMILY

User: My mother takes care of me.

ELIZA: WHO ELSE IN YOU FAMILY TAKES CARE OF YOU

User: My father.

ELIZA: YOUR FATHER

User: You are like my father in some ways.

Weizenbaum, J. (1966). ELIZA - a computer program for the study of natural language communication between man and machine. Communications of the ACM, 9(1), 36-45.

Fields of NLP

- Relation Extraction
- Word Sense Disambiguation
- Machine Translation
- Dependency Parsing
- Transfer Learning
- Text Generation
- Dialogue Systems
- …

Different Tasks Require Different Models

Various approaches to modeling natural language

- State Machines
- Probabilistic Models
- Rule Systems
- Logic
- Vector-Space Models

Highly task dependent

Regular Expressions

Language for specifying text search strings

Used in every computer language, word processor, and text processor tools

Let‘s you define a **pattern** to search for

Core of the ELIZA chat bot

Very helpful & you should learn this!

- https://www.regular-expressions.info/
- https://regex101.com/

Regular Expression	Example Patterns Matched
`/pie/`	„apple pie“ „pieface“ „piece“
`/\bpie\b/`	„apple pie“
`/pie[cf]/`	„pieface“ „piece“
`/^(?!Apple)[A-Za-z]+/`	„PCs are great“ „Blueberry pie is my favorite“ „Why don‘t you like apples?“

The Data

Usually, NLP deals with (large) collections of documents

Corpus: collection of documents or a subset thereof

(documents, sub-documents, paragraphs…)

Dictionary: unique different words that appear in the corpus

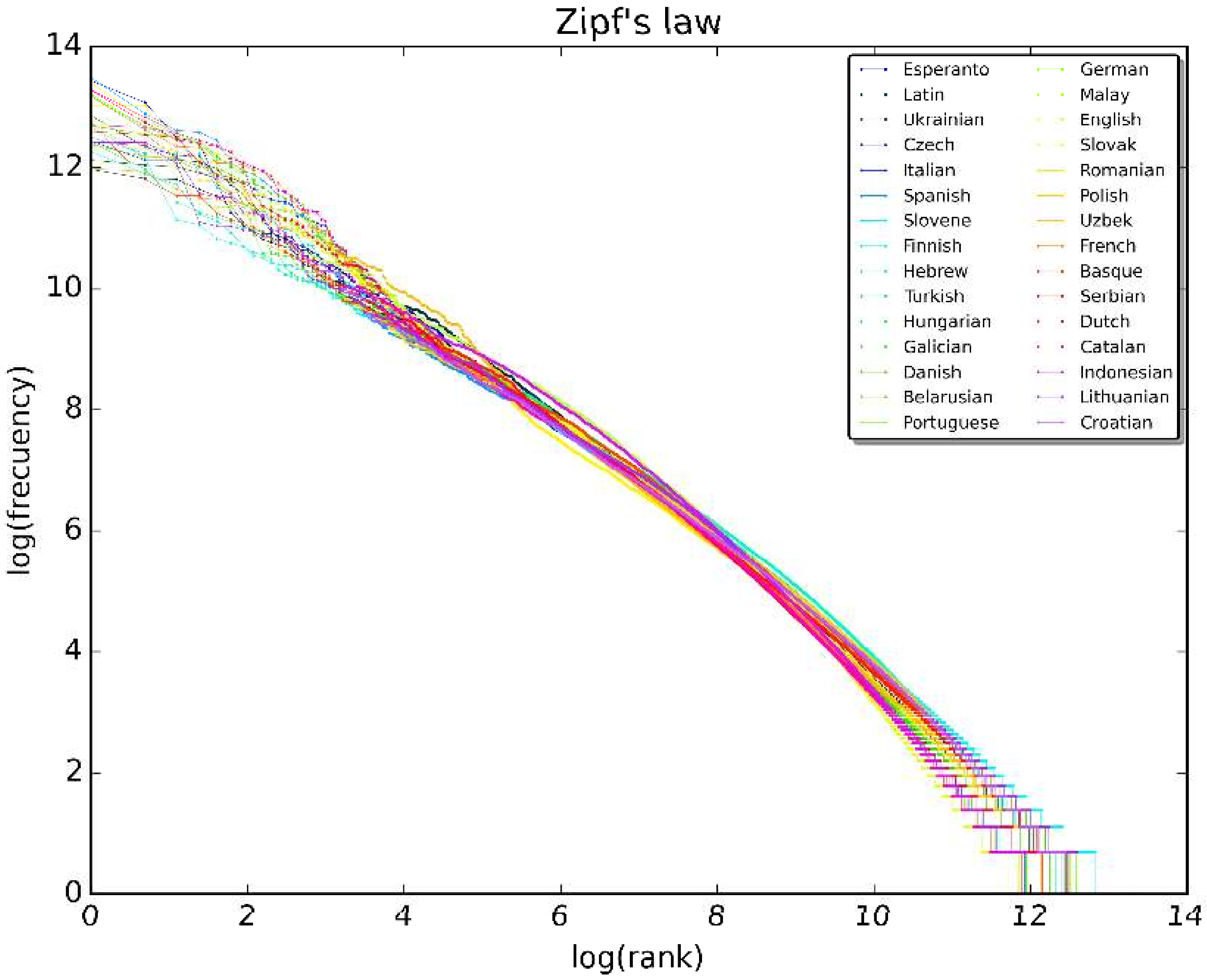
Characteristics of the data: Zipf‘s law

Zipf‘s law:

Data studied in the physical and social sciences follows an inverse relation in the rank-frequency distribution

That means:

- Very few words are responsible for the largest proportion of a written text
- Implication: infrequent words are more important for the message of a text



https://commons.wikimedia.org/wiki/File:Zipf_30wiki_en_labels.png

Preprocessing is crucial

Preprocessing means filtering and altering parts of the corpus in order to improve analysis results

Main goal of preprocessing:
Noise removal

Pipeline on the right:

- Starts with a corpus & produces (possibly) a (sub-)corpus of cleaned documents
- Bold steps: obligatory

- **Removal of irrelevant information**
- **Removal of empty documents**
- **Removal of identical documents**
- **Tokenization**
- Spellchecking
- Removal of repeating characters
- Replacing contractions
- Replacing words with synonyms
- Replacing words with antonyms
- **Lower-case words**
- **Removal of stopwords**
- Lemmatization
- Stemming
- Replacing words with hyperonmys
- Removal of unique tokens

Preprocessing examples

Lowercasing:

- Very simple
- Helps with sparsity issues

Raw	Lowercased
Germany GermAny GERMANY	germany
Airplane AIRPLANE AiRpLaNe	airplane

Stemming:

- Chop off (hopefully) inflection parts of words
- Helps with sparsity issues and standardizing vocabulary

Raw	Stemmed
connect connected connection connections connects	connect
trouble troubled troubles	troubl
troublesome	troublesom

Lemmatization:

- Map words to their root form
- „Sophisticated stemming“
- Requires (mostly) a dictionary
- Often costly

Raw	Lemmatized
trouble troubled troubles troubling	trouble
goose geese	goose

M.F. Porter, 1980, An algorithm for suffix stripping, Program, 14(3) pp 130−137.
https://www.kdnuggets.com/2019/04/text-preprocessing-nlp-machine-learning.html

Preprocessing examples

Normalization:

- Effective
- Highly depending on the type of texts
- Not trivial

Raw	Normalized
2moro 2mrrw 2morrow tomrw	tomorrow
b4	before
otw	on the way
:) :-) ;-) ☺	smile

Further noise removal:

- Remove interfering characters, digits, or pieces of text
- Effective
- Noise: Punctuation, numbers, special characters, source code, header information, domain specific keywords…

Raw	Stemmed	Cleaned	Cleaned & Stemmed
..trouble..	..trouble..	trouble	troubl
trouble<	trouble<	trouble	troubl
trouble!	trouble!	trouble	troubl
<a>trouble</a>	<a>trouble</a>	trouble	troubl
1.trouble	1.troubl	trouble	troubl

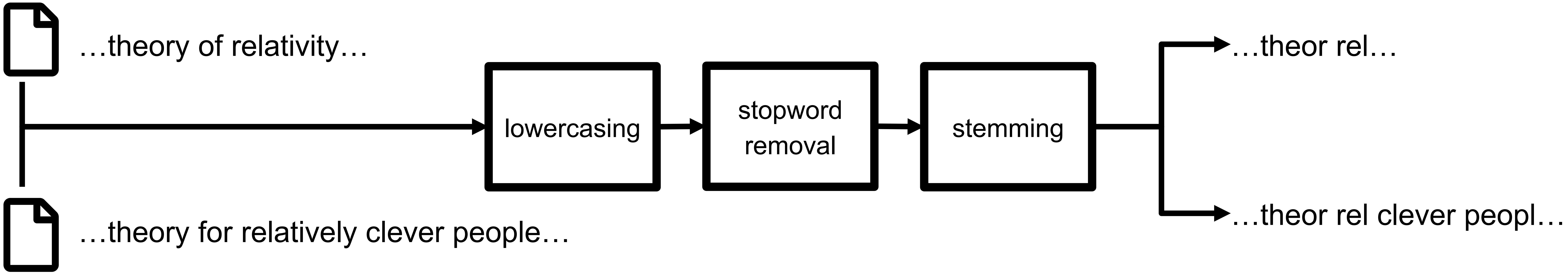
https://www.kdnuggets.com/2019/04/text-preprocessing-nlp-machine-learning.html

…but also highly task-dependent!

What is considered „noise“ depends on the task!

Preprocessing steps showing effective results for one task may be unhelpful in another

Suppose you want to keep „theory of relativity“ as a fixed token inside your corpus



Problem here:
different input patterns map to the same output feature „theor rel“

https://www.kdnuggets.com/2019/04/text-preprocessing-nlp-machine-learning.html

Bags of words

Documents are bags of words

Bag-of-words model is a simplifying representation of documents

Disregards word order

But keeps multiplicity.

Mary is quicker than John

```
BOW = {"Mary":1, "is":1,
       "quicker":1, "than":1,
       "John":1}
```

John is quicker than Mary

```
BOW = {"Mary":1, "is":1,
       "quicker":1, "than":1,
       "John":1}
```

Intuition:

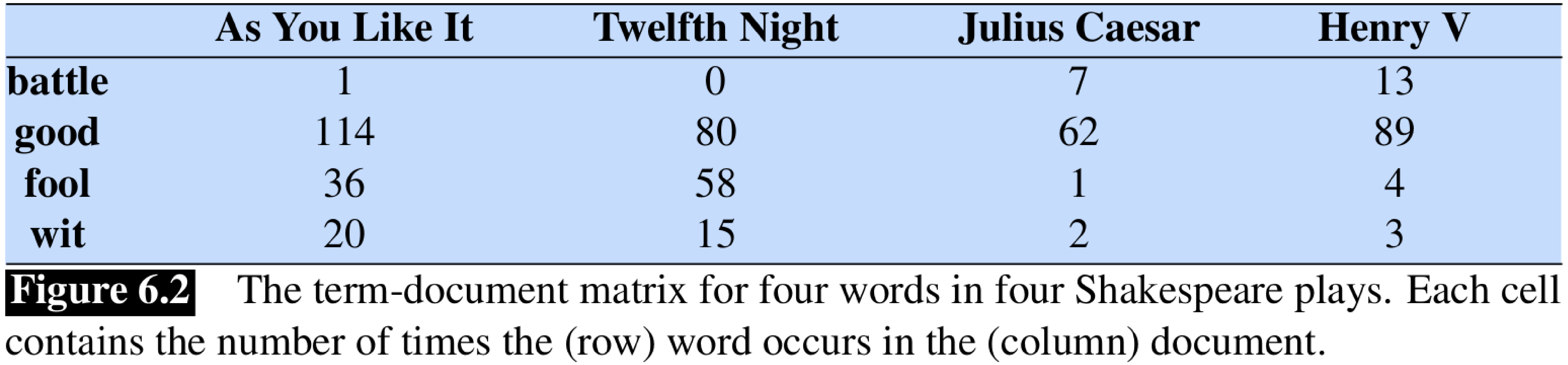
Documents with similar bag of words representations are similar in content

https://commons.wikimedia.org/wiki/File:The_original_1920_English_publication_of_the_paper..jpg
https://www.marxists.org/reference/archive/einstein/works/1910s/relative/index.htm

Document Term Matrix

Describes the frequency of terms (unigrams) that occur in a collection of documents

Rows correspond to documents, columns correspond to terms



	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 inverse is also common:

In a term-document matrix, rows contain terms & columns contain documents

Not to be confused with a document feature matrix

(which contains more than terms, e.g. n-grams, compound tokens, …)

Jurafsky, D., & Martin, J. H. (2021, draft). Speech and language processing

Application of bag of words paradigm: Topic Modeling

- Statistical approach to discover abstract “topics” in text documents
- Topics are latent mechanisms influencing co-occurrence of words
- A lot of variances to the original algorithm known

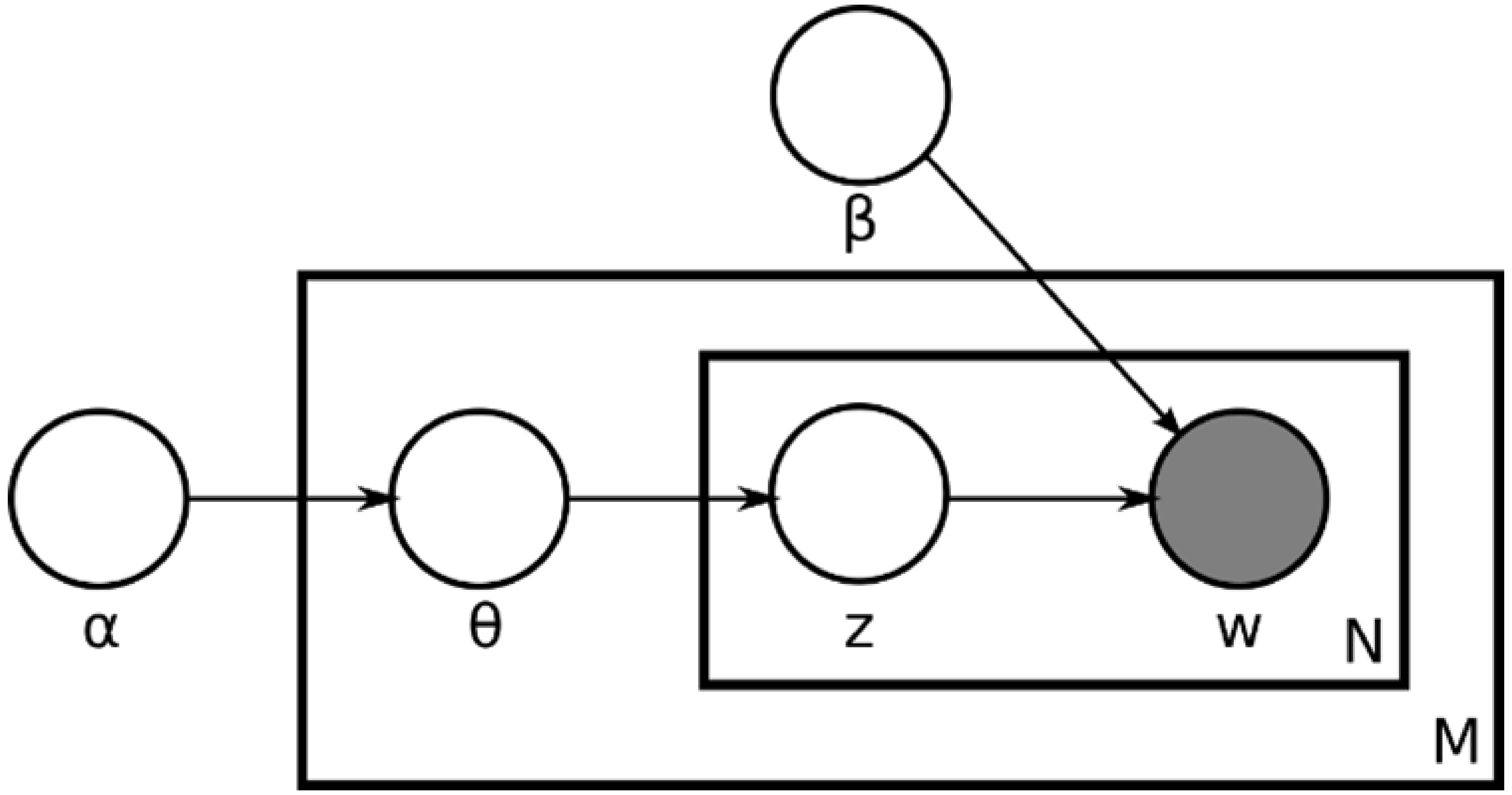
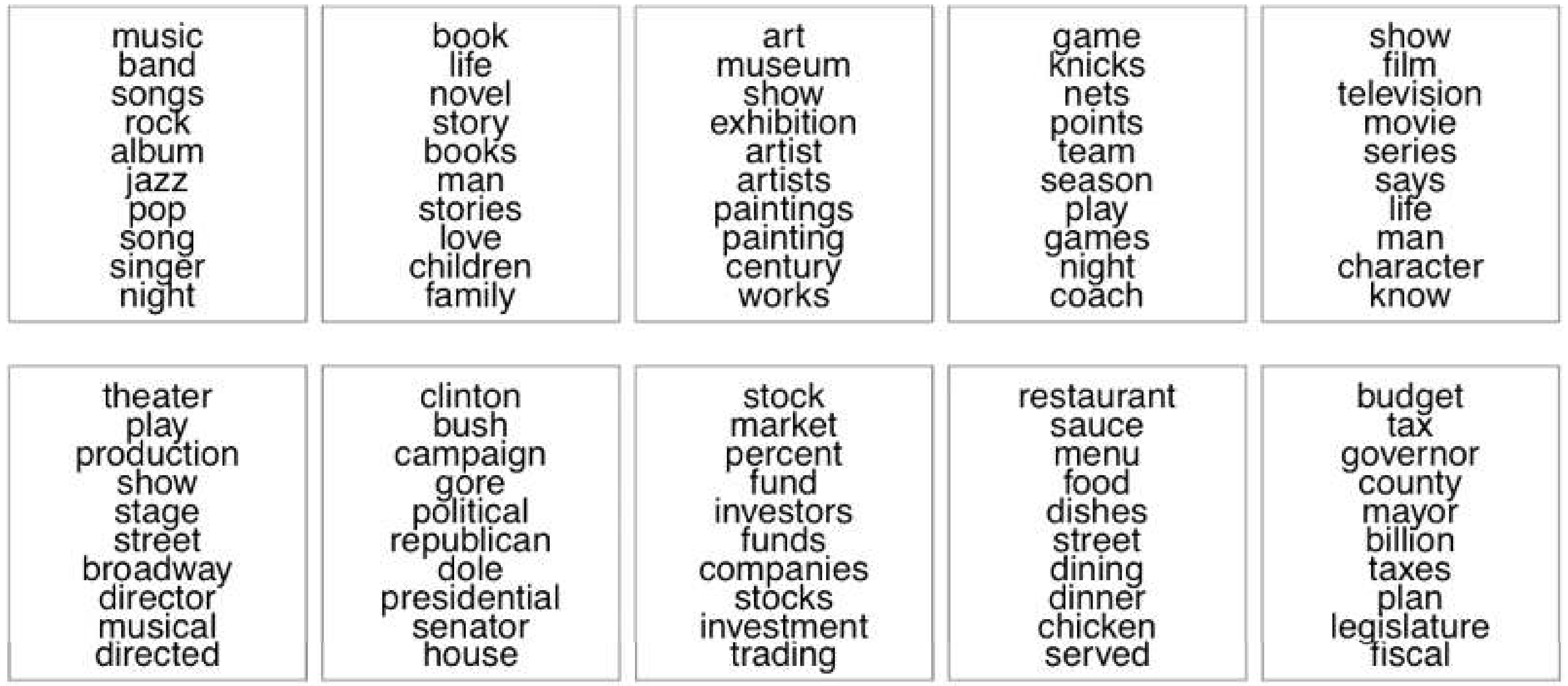


Plate-notation of Latent-Dirichlet-Allocation



music	book	art	game	show
band	life	museum	knicks	film
songs	novel	show	nets	television
rock	story	exhibition	points	movie
album	books	artist	team	series
jazz	man	artists	season	says
pop	stories	paintings	play	life
song	love	painting	games	man
singer	children	century	night	character
night	family	works	coach	know

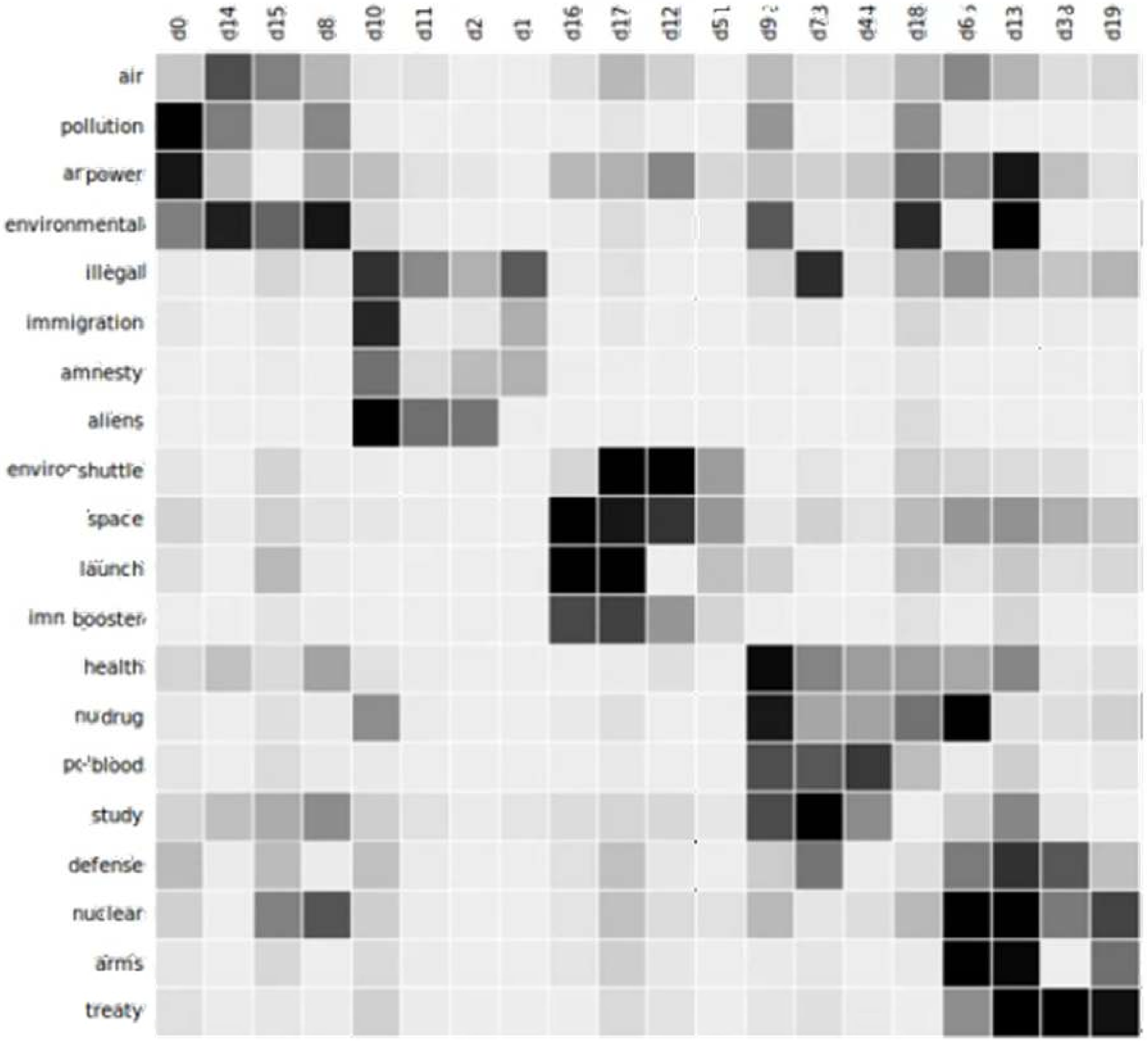
theater	clinton	stock	restaurant	budget
play	bush	market	sauce	tax
production	campaign	percent	menu	governor
show	gore	fund	food	county
stage	political	investors	dishes	mayor
street	republican	funds	street	billion
broadway	dole	companies	dining	taxes
director	presidential	stocks	dinner	plan
musical	senator	investment	chicken	legislature
directed	house	trading	served	fiscal

“Some of the topics found by analyzing 1.8 million articles from the New York Times. Each panel illustrates a set of tightly co-occurring terms in the collection.” (Blei, 2012)

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent Dirichlet Allocation. Journal of Machine Learning Research, 3(Jan), 993-1022.
Blei, D., & Lafferty, J. (2005). Correlated topic models. Advances in neural information processing systems, 18, 147.
Blei, D. M. (2012). Topic modeling and digital humanities. Journal of Digital Humanities, 2(1), 8-11.

Topic Modeling

- Latent Dirichlet Allocation (LDA) is a data mining algorithm that infers latent mechanisms by clustering co-occurrences of words in documents
- Common technique to summarize/”access” large corpora
- Validating such models requires both a deep statistical understanding as well as lots of qualitative work

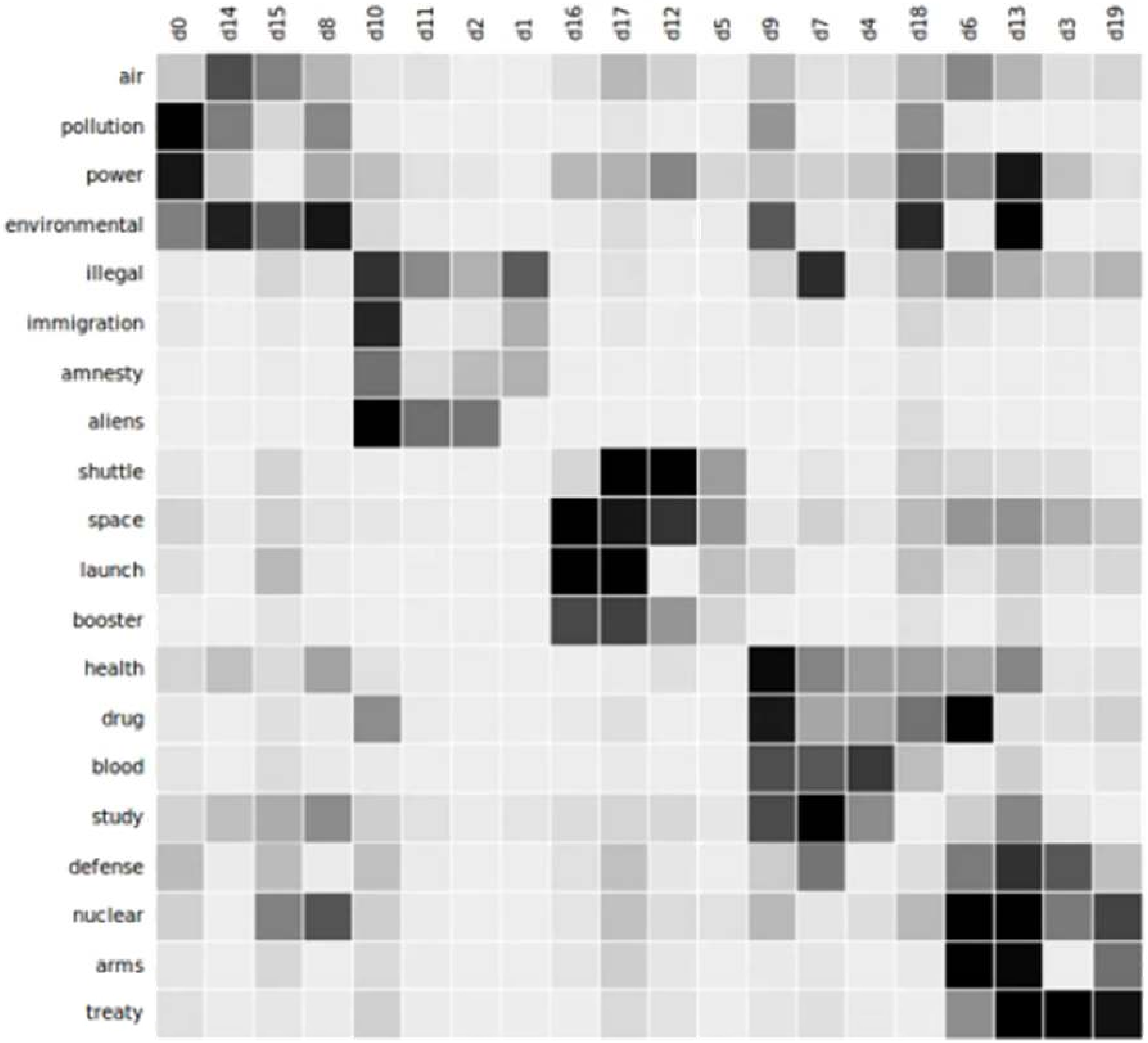


Document-Term-Matrix sorted by co-occurrences of words in documents
https://en.wikipedia.org/wiki/Topic_model

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent Dirichlet Allocation. Journal of Machine Learning Research, 3.
Blei, D., & Lafferty, J. (2005). Correlated topic models. Advances in neural information processing systems, 18, 147.

Topic Modeling

- Topic 1:
  air, pollution, power, environmental
- Topic 2:
  illegal, immigration, amnesty, aliens
- Topic 3:
  shuttle, space, launch, booster
- Topic 4:
  health, drug, blood, study
- Topic 5:
  defense, nuclear, arms, treaty



Document-Term-Matrix sorted by co-occurrences of words in documents
https://en.wikipedia.org/wiki/Topic_model

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent Dirichlet Allocation. Journal of Machine Learning Research, 3.
Blei, D., & Lafferty, J. (2005). Correlated topic models. Advances in neural information processing systems, 18, 147.

If order is important…

Language Models

Language models consider sequences of words, i.e. they maintain a sense of word order

Language models assign probabilities to sequences of words

What word will likely follow?

```
Please turn your homework …
```

Which sequence has a higher probability for appearing in a text?

```
all of a sudden I notice three guys standing on the sidewalk
on guys all I of notice sidewalk three a sudden standing the
```

Language models are useful for

- Speech recognition
- Spelling correction
- Grammatical error correction
- Machine translation
- …

n-gram language models

Simplest language model

Sequence of *n* words:

- 2-gram (**bigram**): „please turn“, „turn your“, „your homework“,…
- 3-gram (**trigram**): „please turn your“, „turn your homework“,…
- …

Trigrams are commonly used.

4-gram or 5-gram models are even better but require a lot more training data.

Side note: large n-gram models require padding with pseudo-words, e.g. at the beginning of sentences

Aim: compute $P(w|h)$ the probability of a word w given some history h

n-gram by example

w: the
h: its water is so transparent that

$$P(the|its\ water\ is\ so\ transparent\ that)$$

Usually, these probabilites are estimated from very large corpora counting the occurences and putting them into relation:

$$\frac{C(its\ water\ is\ so\ transparent\ that\ the)}{C(its\ water\ is\ so\ transparent\ that)}$$

Even slight variations in the text might yield counts of zero even in Internet-scale corpora,
for example „Red Main‘s water is so transparent that…“.
So, instead of calculating the word probability using the complete history

$$P(w_n|w_{1:n-1})$$

we approximate the history by just the last *n* words:

$$P(w_n|w_{1:n-1}) \approx P(w_n|w_{n-N+1:n-1})$$

For the bigram model this means to use only the previous word:

$$P(the|that)$$

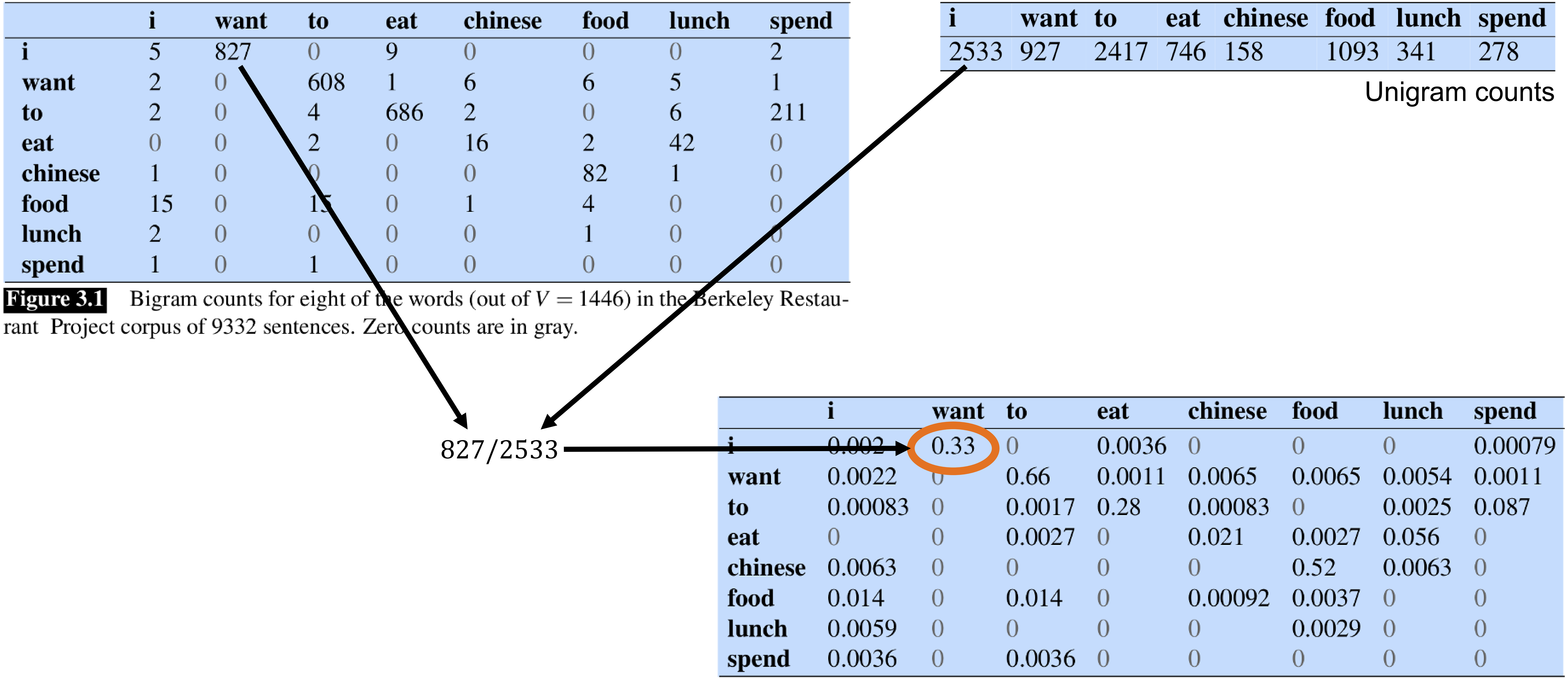
This is a **Markov** assumption: we can predict the probability of some future without looking too far into the past

Using bigrams, we can compute the probability of a complete word sentence by using the **chain rule of probability**:

$$P(w_{1:n}) \approx \prod_{k=1}^{n} P(w_k|w_{k-1})$$

Jurafsky, D., & Martin, J. H. (2021, draft). Speech and language processing

n-gram by example



	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

Figure 3.1 Bigram counts for eight of the words (out of $V = 1446$) in the Berkeley Restaurant Project corpus of 9332 sentences. Zero counts are in gray.

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

Unigram counts

827/2533

	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

Figure 3.2 Bigram probabilities for eight words in the Berkeley Restaurant Project corpus of 9332 sentences. Zero probabilities are in gray.

Jurafsky, D., & Martin, J. H. (2021, draft). Speech and language processing

Applications of n-gram language models

Widely used in NLP

Application areas include

- Speech recognition
- Language identification
- Machine translation
- OCR
- …

Even applied in extracting features from image data

Further variances: skip-grams

What about the context?

> You shall know a word
> by the company it keeps

John Rupert Firth, 1957

Distributional Hypothesis

Words that are used and occur in the same contexts tend to purport similar meaning

Suggestion of the hypothesis:

Semantic of words has an effect on their distribution in language use

By observing distribution of words, we might infer something about the meaning

Suppose you don‘t know the meaning of the word *ongchoi* but you see it in these contexts

(6.1) Ongchoi is delicious sauteed with garlic.
(6.2) Ongchoi is superb over rice.
(6.3) ...ongchoi leaves with salty sauces...

…and suppose that you had seen many of the context words in other contexts

(6.4) ...spinach sauteed with garlic over rice...
(6.5) ...chard stems and leaves are delicious...
(6.6) ...collard greens and other salty leafy greens

Can you infer something about *ongchoi* from this?

Firth, John R. "A synopsis of linguistic theory, 1930-1955." Studies in linguistic analysis (1957).

Vector embeddings

Words are represented as a point in a multidimensional semantic space

Semantic spaces are derived from the distributions of word neighbors, i.e. contexts

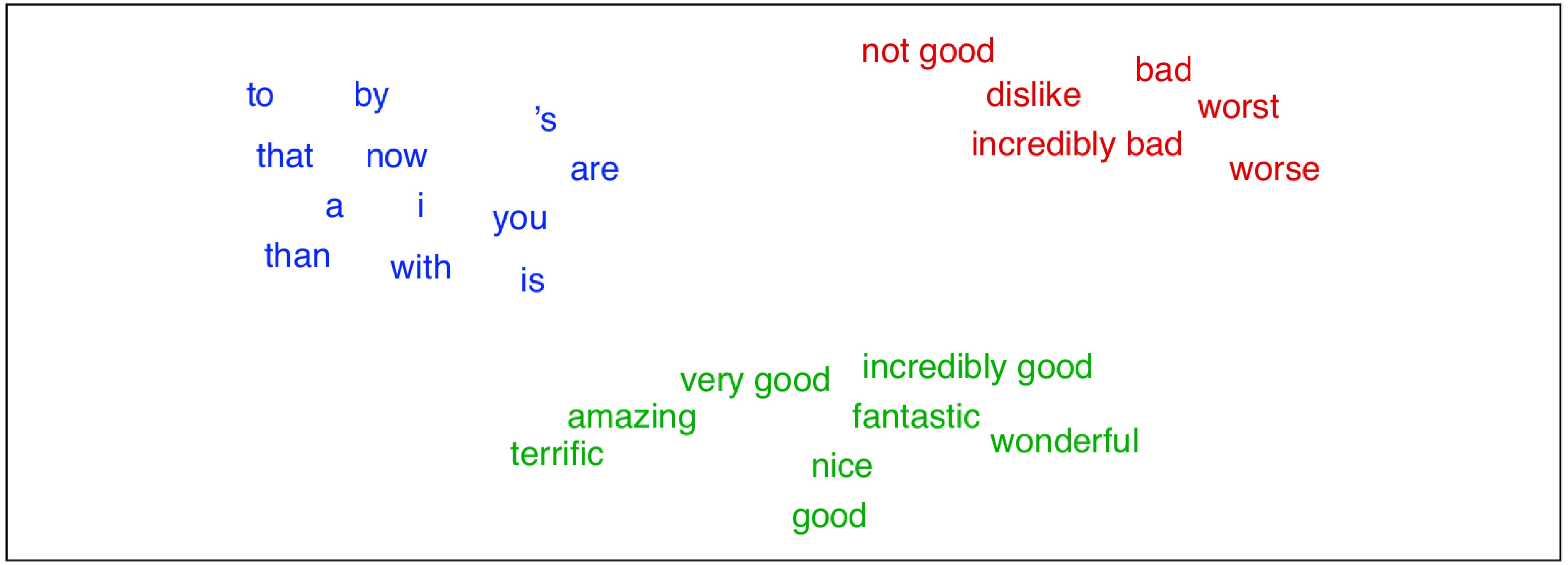
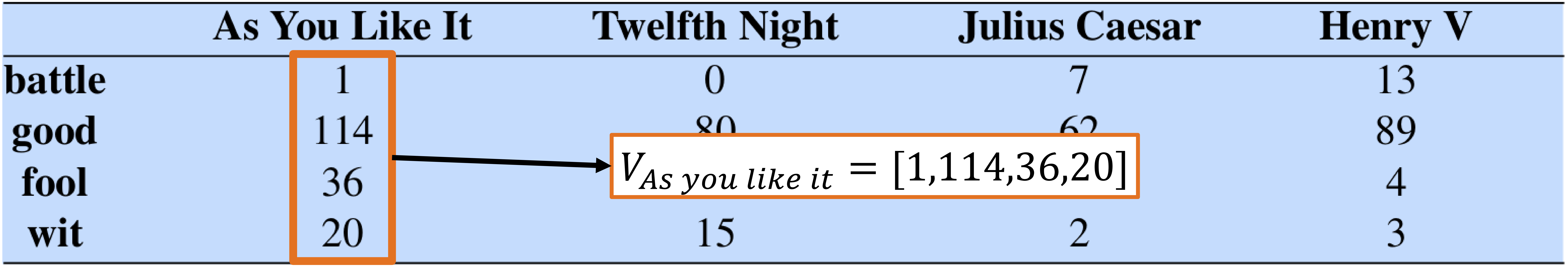


Figure 6.1 A two-dimensional (t-SNE) projection of embeddings for some words and phrases, showing that words with similar meanings are nearby in space. The original 60-dimensional embeddings were trained for sentiment analysis. Simplified from Li et al. (2015) with colors added for explanation.

Jurafsky, D., & Martin, J. H. (2021, draft). Speech and language processing

From Text to Vectors

Starting from term-document matrix, document vectors can be obtained from columns



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

$V_{As\ you\ like\ it} = [1,114,36,20]$

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.

A vector space is a collection of vectors, characterized by their dimension.

Dimension is usually the vocabulary size denoted by $|V|$

Jurafsky, D., & Martin, J. H. (2021, draft). Speech and language processing

From Text to Vectors

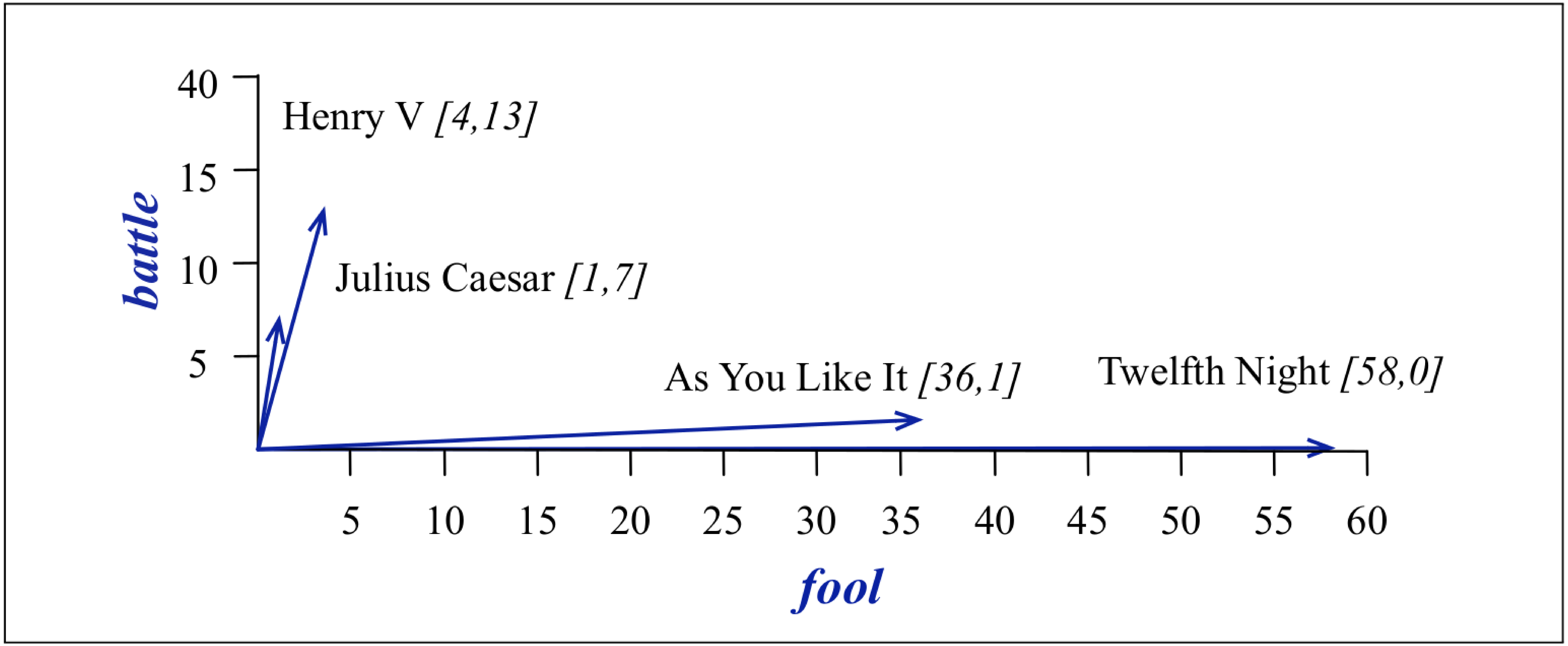


Figure 6.4 A spatial visualization of the document vectors for the four Shakespeare play documents, showing just two of the dimensions, corresponding to the words *battle* and *fool*. The comedies have high values for the *fool* dimension and low values for the *battle* dimension.

Jurafsky, D., & Martin, J. H. (2021, draft). Speech and language processing

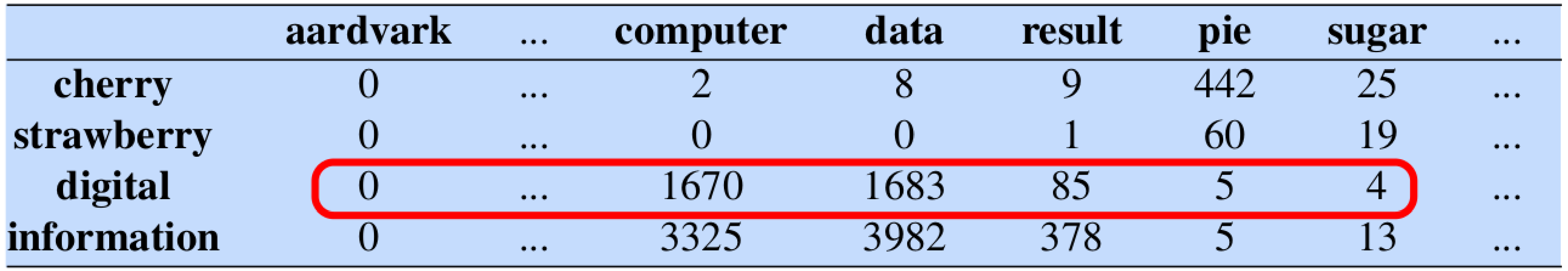
Words as vectors

Based on the term-term matrix or the term-context matrix we can derive word vectors

Term-term matrices have dimensionality $|V| \times |V|$

Each cell records how often a target word (row) and a context word (colum) co-occur

is traditionally followed by	**cherry**	pie, a traditional dessert
often mixed, such as	**strawberry**	rhubarb pie. Apple pie
computer peripherals and personal	**digital**	assistants. These devices usually
a computer. This includes	**information**	available on the internet



	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	...

Figure 6.6 Co-occurrence vectors for four words in the Wikipedia corpus, showing six of the dimensions (hand-picked for pedagogical purposes). The vector for *digital* is outlined in red. Note that a real vector would have vastly more dimensions and thus be much sparser.

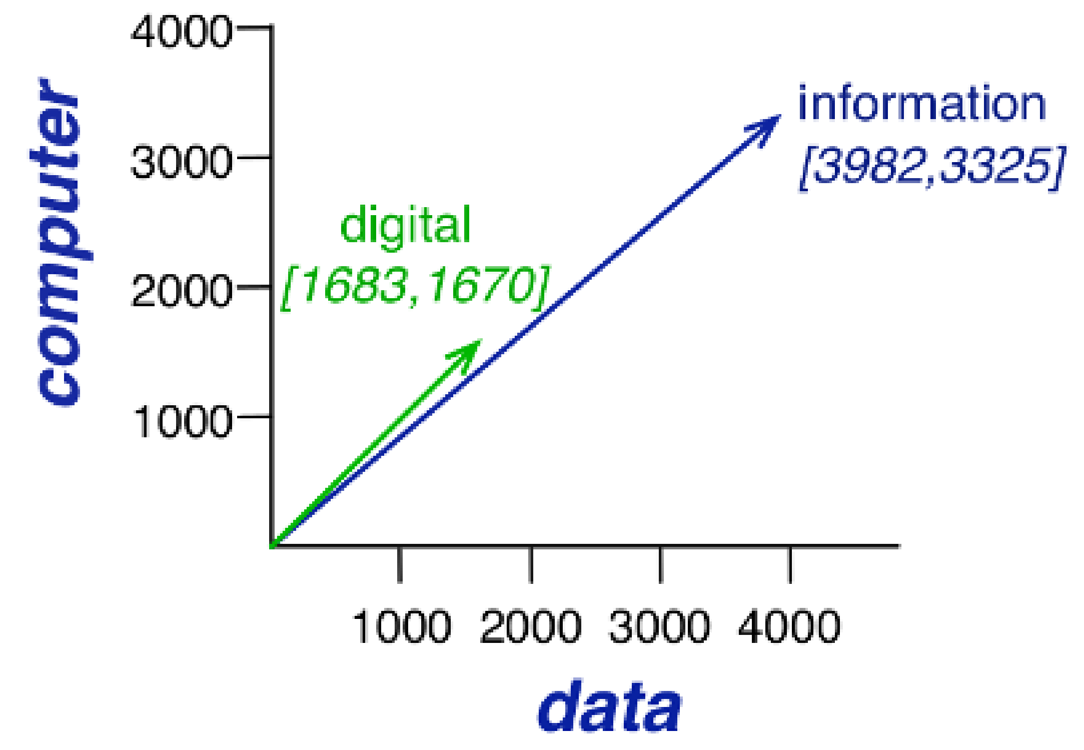


Figure 6.7 A spatial visualization of word vectors for *digital* and *information*, showing just two of the dimensions, corresponding to the words *data* and *computer*.

Jurafsky, D., & Martin, J. H. (2021, draft). Speech and language processing

Similarities between words

Based on word vectors we can obtain Cosine similarity between two vectors

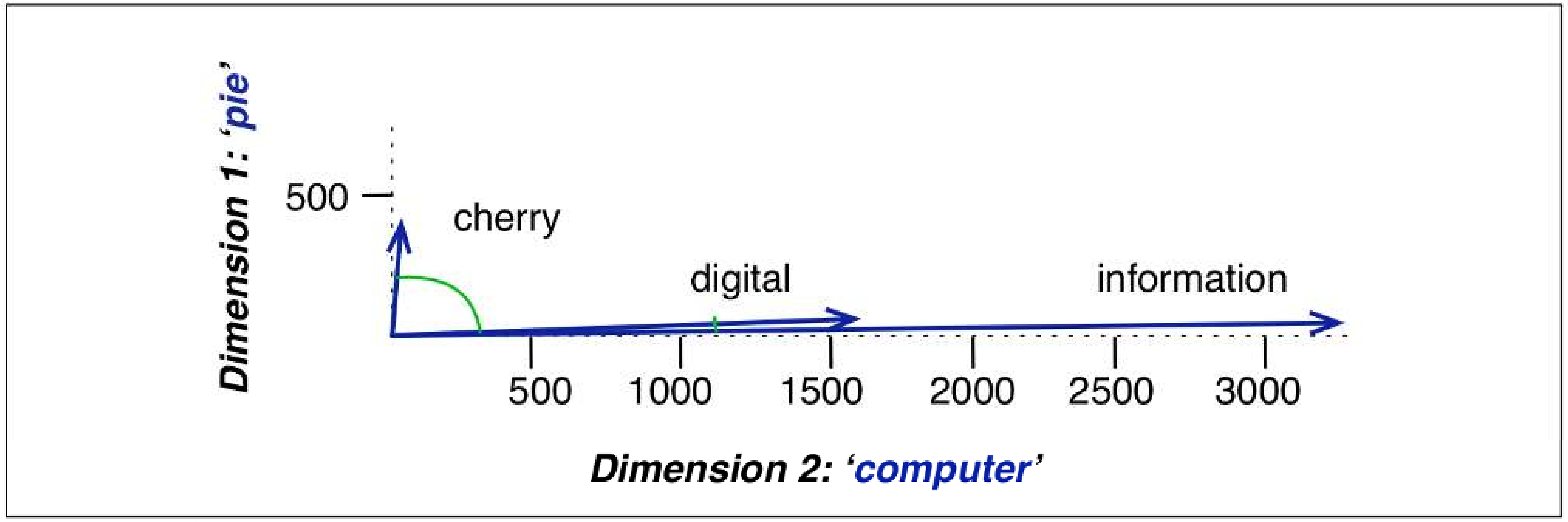


Figure 6.8 A (rough) graphical demonstration of cosine similarity, showing vectors for three words (*cherry*, *digital*, and *information*) in the two dimensional space defined by counts of the words *computer* and *pie* nearby. The figure doesn't show the cosine, but it highlights the angles; note that the angle between *digital* and *information* is smaller than the angle between *cherry* and *information*. When two vectors are more similar, the cosine is larger but the angle is smaller; the cosine has its maximum (1) when the angle between two vectors is smallest (0°); the cosine of all other angles is less than 1.

However, using the raw frequency leads to skewed results that aren't discriminative

Jurafsky, D., & Martin, J. H. (2021, draft). Speech and language processing

Ranking of words: TF-IDF

A numerical statistic reflecting how important a word is to a document of a corpus.

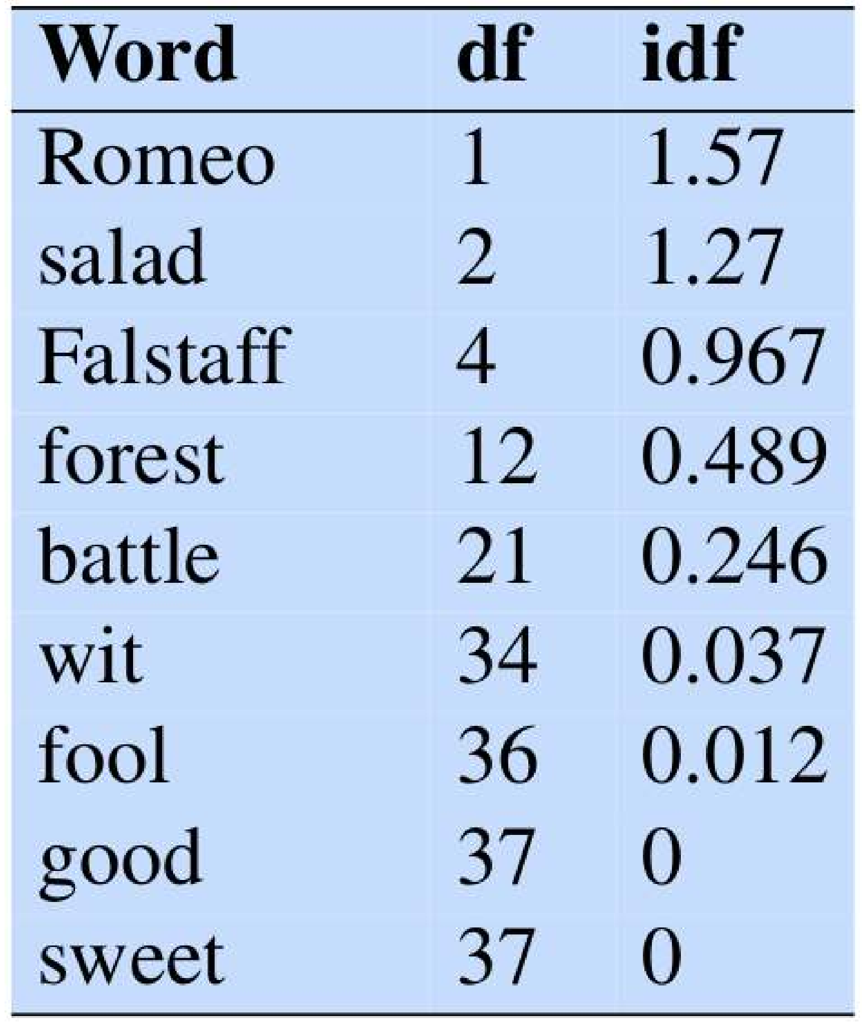
Assumes that frequent terms are less informative (remember Zipf‘s law?)

Very common (and probably the best) weighting scheme in information retrieval

$$tfidf(t,d,D) = tf(t,d) \times idf(t,D)$$

Manning, C. D., Raghavan, P., & Schütze, H. (2008). Introduction to Information Retrieval. Cambridge: Cambridge University Press.

TF-IDF



Word	df	idf
Romeo	1	1.57
salad	2	1.27
Falstaff	4	0.967
forest	12	0.489
battle	21	0.246
wit	34	0.037
fool	36	0.012
good	37	0
sweet	37	0

Term Frequency

$$tf(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}$$

Inverse Document Frequency

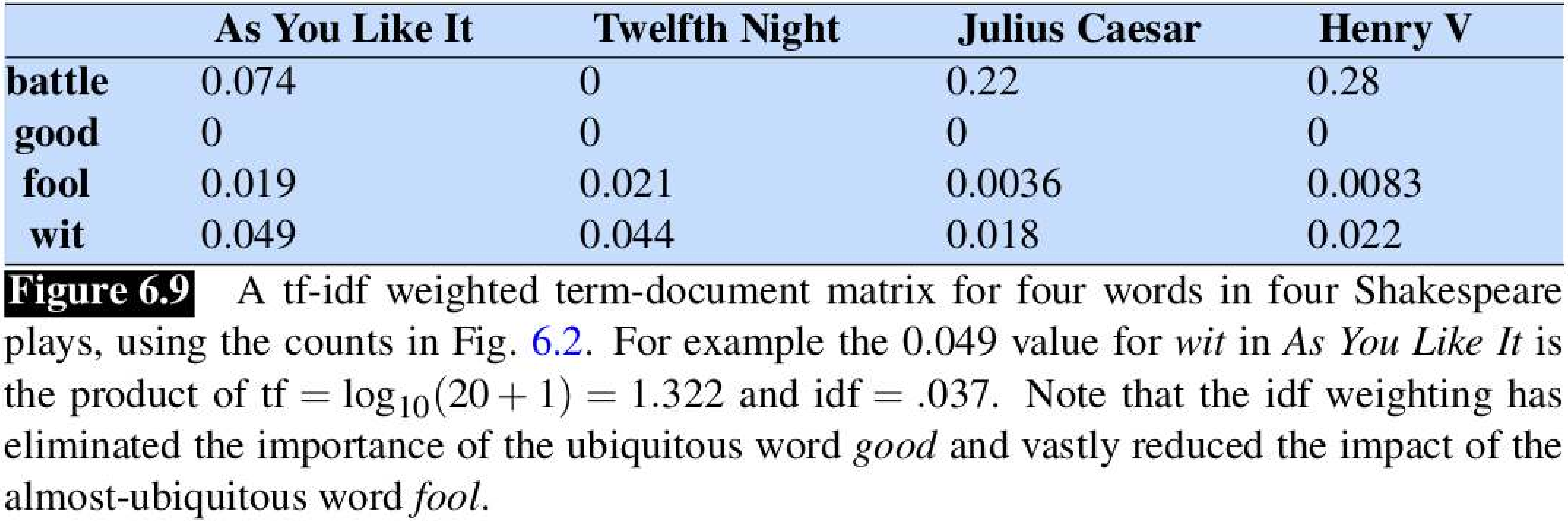
$$idf(t,D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

$$tfidf(t,d,D) = tf(t,d) \times idf(t,D)$$

$f_{t,d}$	Frequency of term t in document d
$\sum_{t' \in d} f_{t',d}$	Number of terms in the document
N	Number of documents in the corpus
$\|\{d \in D : t \in d\}\|$	Number of documents d that contain the term t

Manning, C. D., Raghavan, P., & Schütze, H. (2008). Introduction to Information Retrieval. Cambridge: Cambridge University Press.

TF-IDF



	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

Figure 6.9 A tf-idf weighted term-document matrix for four words in four Shakespeare plays, using the counts in Fig. 6.2. For example the 0.049 value for *wit* in *As You Like It* is the product of $\text{tf} = \log_{10}(20+1) = 1.322$ and $\text{idf} = .037$. Note that the idf weighting has eliminated the importance of the ubiquitous word *good* and vastly reduced the impact of the almost-ubiquitous word *fool*.

Application of TF-IDF

- Recommender systems

  Requires: a profile / a query-document & a database of documents with known TF-IDF scores

  Identify documents in a database that contain the same words with a similar importance

  Obtain TF-IDF scores for words in the query document

  Measure cosine similarity between TF-IDF vector of query document and TF-IDF vectors of the database

  Use closest documents from the database as recommendations

- Search

  Requires: a search query & a database of documents with known TF-IDF scores

  Obtain documents for which terms of search query have a noticeable TF-IDF score

- Automatic stopword detection

  Requires: TF-IDF scores for a document

  Consider all words below a certain threshold as stopwords

Word embeddings

Word embeddings

Previous vector representations of words are large and sparse
Each word-vector has $|V|$ or $|D|$ dimensions most of which are zero

Powerful word representation: short dense **embedding vectors**

Embeddings have a fixed number of dimensions d (usually 50-1000)

Dimensions don‘t correspond to presence or absence of individual words anymore
They do a better job of capturing synonymy of words

Static embeddings

One fixed embedding for each word in the vocabulary

word2vec

Dynamic embeddings

Different embeddings for words in different contexts

BERT

word2vec: skip-gram with negative sampling

Intuition:

Train a binary classifier answering „is word w likely to show up near, say, *apricot*?“

Don‘t actually predict anything later, but use the classifier *weights* as word embeddings

Self-supervising:

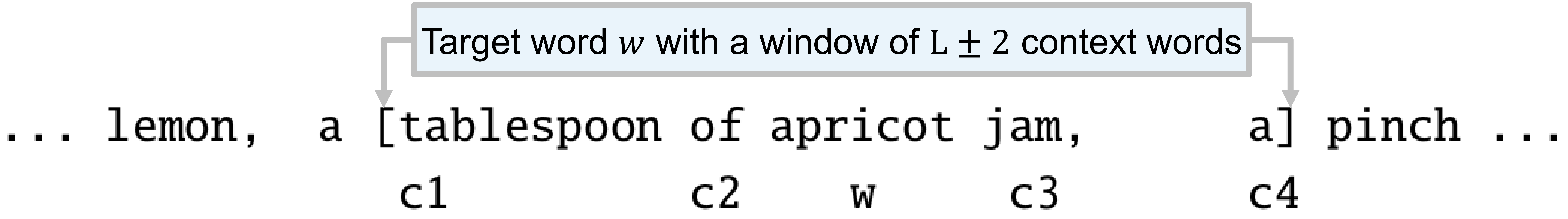
Use running text as implicitly supervised training data

Algorithm:

1. Treat a target word and a neighboring context word as positive examples
2. Randomly sample other words in the lexicon to get negative samples
3. Use logistic regression to train a classifier to distinguish those two cases
4. Use the learned weights as the embeddings

Mikolov, Tomas, et al. "Efficient estimation of word representations in vector space." arXiv preprint arXiv:1301.3781 (2013).
Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality." Advances in neural information processing systems. 2013.

word2vec in detail



Target word w with a window of $L \pm 2$ context words

`... lemon, a [tablespoon of apricot jam, a] pinch ...`

`c1 c2 w c3 c4`

Our goal: knowing the probability for a pair of target word w with a candidate context word c that c is a real context word:

$$P(+|w, c)$$

Therefore, we need embeddings for each target word and context word in the vocabulary.
From the L window we get positive training instances. And for each of these we sample k negative samples (here $k = 2$).
Noise words are not chosen randomly but according to some weight.

positive examples +

w	c_{pos}
apricot	tablespoon
apricot	of
apricot	jam
apricot	a

negative examples -

w	c_{neg}	w	c_{neg}
apricot	aardvark	apricot	seven
apricot	my	apricot	forever
apricot	where	apricot	dear
apricot	coaxial	apricot	if

1. Maximize the similarity of the (w, c_{pos}) pairs of positive examples
2. Minimize the similarity of the (w, c_{neg}) pairs of negative examples

word2vec in detail

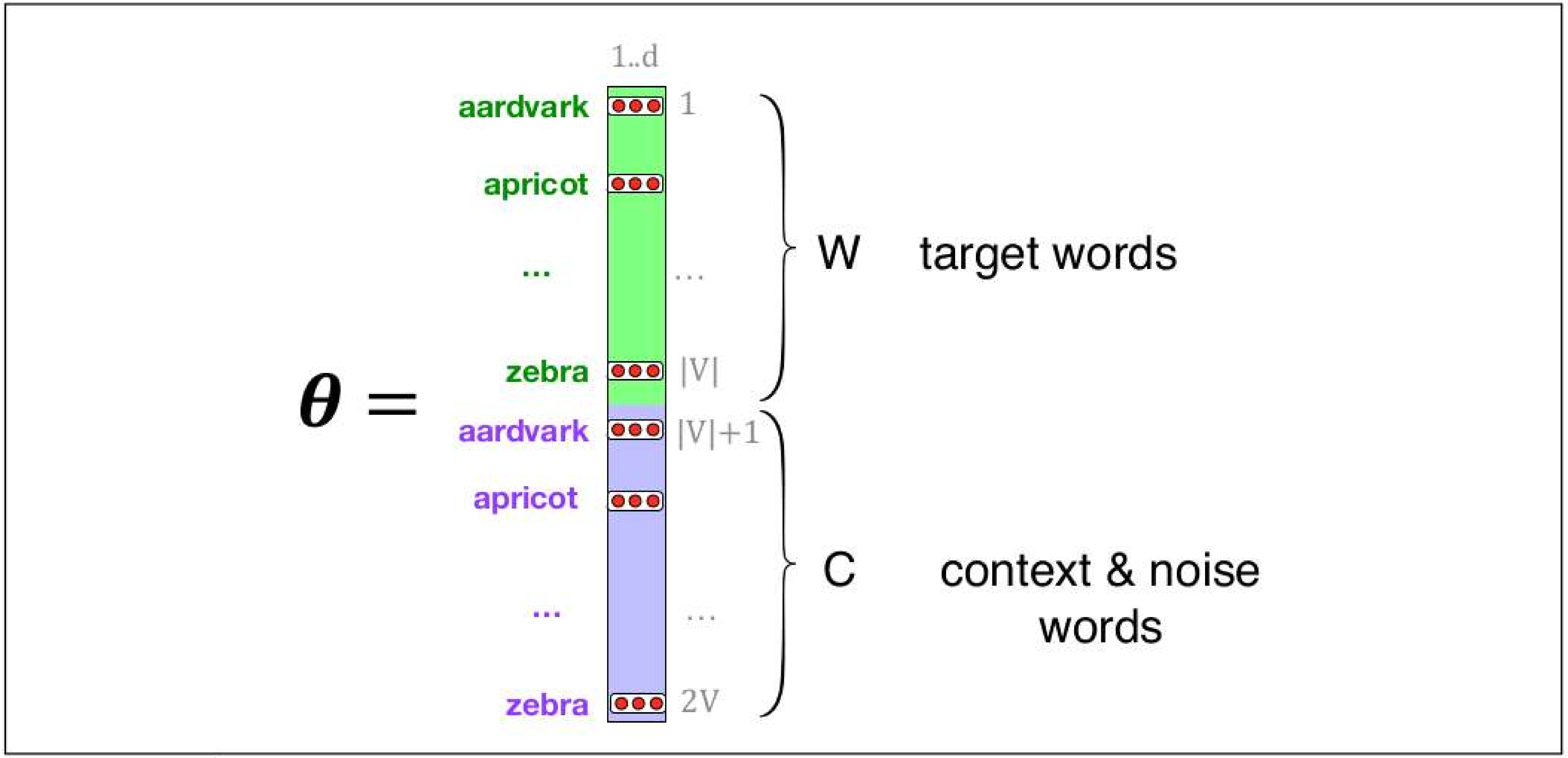


Figure 6.13 The embeddings learned by the skipgram model. The algorithm stores two embeddings for each word, the target embedding (sometimes called the input embedding) and the context embedding (sometimes called the output embedding). The parameter θ that the algorithm learns is thus a matrix of $2|V|$ vectors, each of dimension d, formed by concatenating two matrices, the target embeddings **W** and the context+noise embeddings **C**.

word2vec in detail

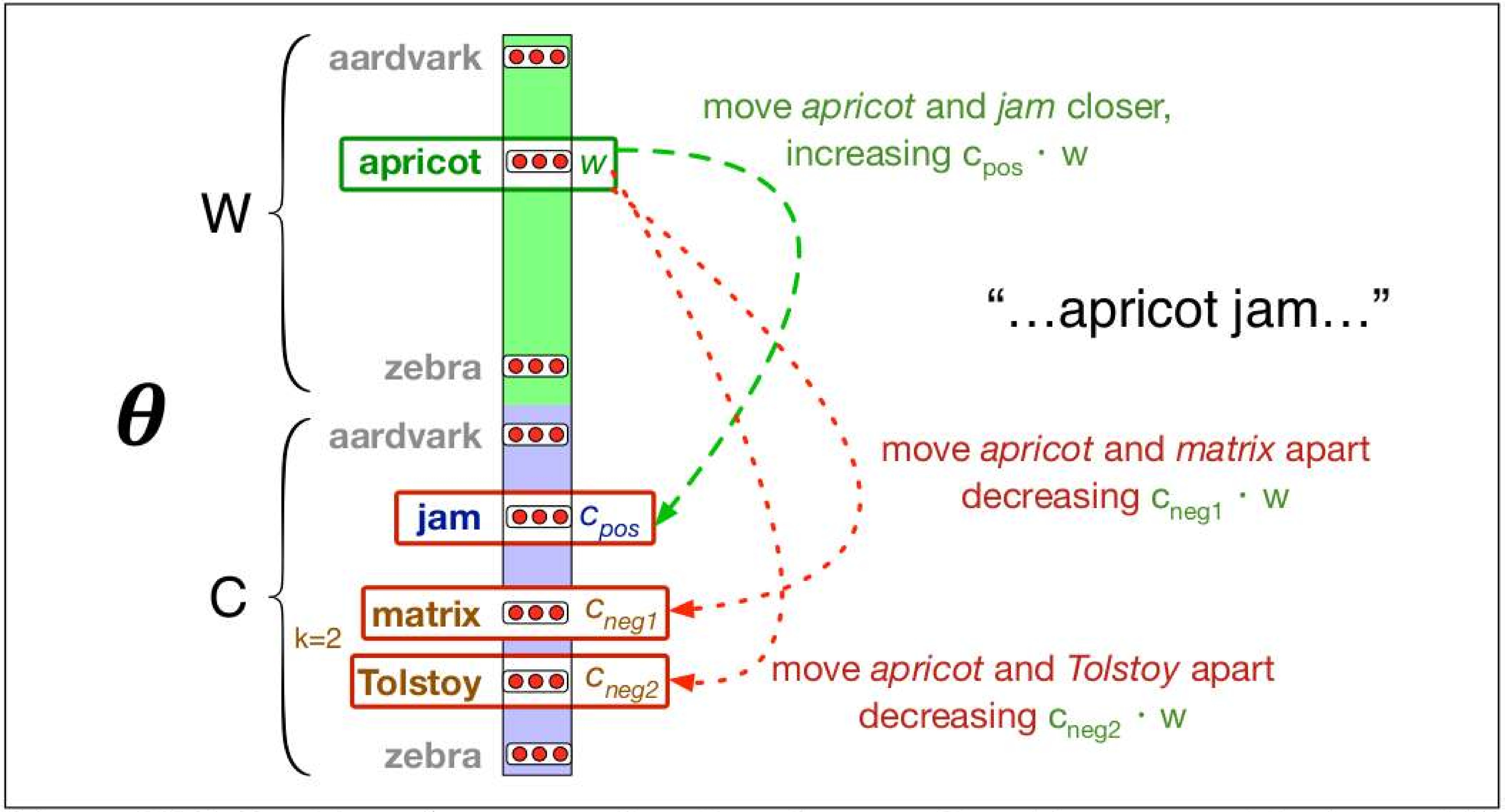
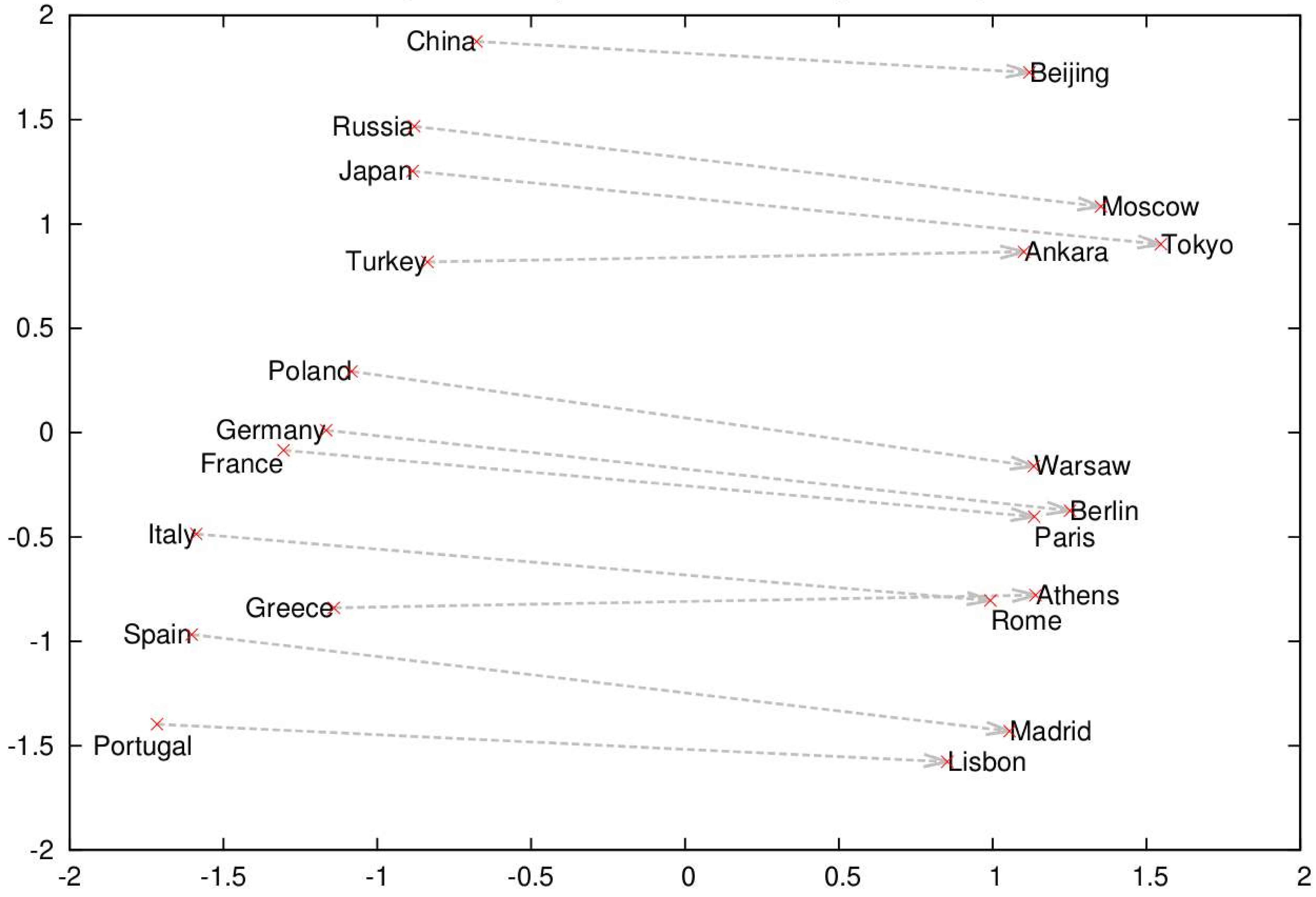

θ
W
aardvark
apricot
w
zebra
C
aardvark
jam
c_{pos}
k=2
matrix
c_{neg1}
Tolstoy
c_{neg2}
zebra
move *apricot* and *jam* closer,
increasing $c_{pos} \cdot w$
"…apricot jam…"
move *apricot* and *matrix* apart
decreasing $c_{neg1} \cdot w$
move *apricot* and *Tolstoy* apart
decreasing $c_{neg2} \cdot w$

Figure 6.14 Intuition of one step of gradient descent. The skip-gram model tries to shift embeddings so the target embeddings (here for *apricot*) are closer to (have a higher dot product with) context embeddings for nearby words (here *jam*) and further from (lower dot product with) context embeddings for noise words that don't occur nearby (here *Tolstoy* and *matrix*).

word2vec: appealing properties



Two-dimensional projection of 1000-dimensional word2vec-vectors of countries and their capital cities.

Interestingly, the vector calculation

$$\overrightarrow{Madrid} - \overrightarrow{Spain} + \overrightarrow{France}$$

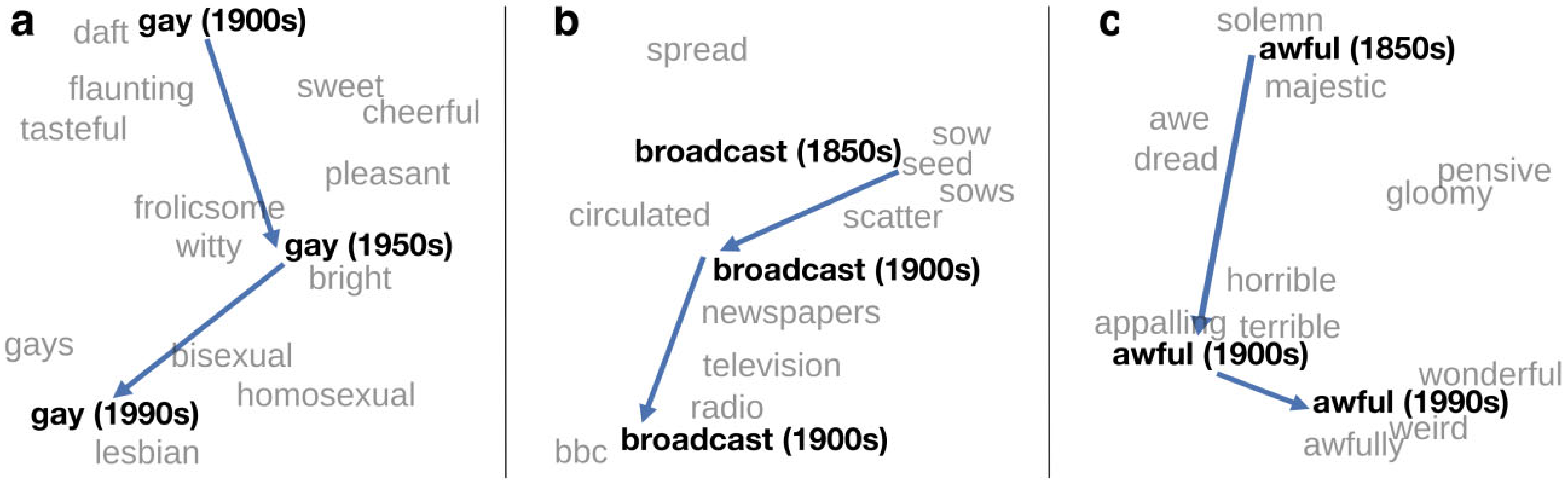
yields a result that is closer to

$$\overrightarrow{Paris}$$

than to any other word vector.

Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality." Advances in neural information processing systems. 2013.

word2vec: appealing properties



A t-SNE visualization of the semantic change of 3 words in English using word2vec vectors. The modern sense of each word, and the grey context words, are computed from the most recent (modern) time-point embedding sapce. Earlier points are computed from earlier historical embedding spaces.

Hamilton, William L. et al. "Diachronic word embeddings reveal statistical laws of semantic change." Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (2016).

Limitations of word2vec

- Bias amplification

  for example: gendered terms become more gendered
- Representational harm

  for example: African-American names showed higher cosine similarity with unpleasent words
- Intrinsic evaluation difficult

  performance is tested on correlating model‘s word similarities between ratings assigned by humans
- Inherent variability

  algorithms may produce different results even from the same dataset,
  and individual documents may strongly impact resulting embeddings

BERT

Contextual embedding:

Each word w will be represented by a different vector each time it appears in a different context

Bidirectional Transformer Encoders:

Model is not based on an incremental, left-to-right processing of inputs but rather looks in both directions

> Predict the missing term given the rest of the sentence:
>
> `Please turn ___ homework in.`

Produces a **pretrained language model** that has great generalization capabilities.
To use these models in other tasks, models are **fine-tuned** by adding small sets of application-specific parameters.

Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (2019).

Limitations

- Machines don‘t understand meaning
- Models often require huge amount of training
- Complex models are difficult to evaluate
- Suffer from various bias

Different models…

What did we see today?

- Vector space model: Bag-of-words
- Probabilistic model: n-grams
- Vector space model: word embeddings

Of course, there‘s more

- Neural network language models
- Graph-based models
- …

Thanks.

mirco.schoenfeld@uni-bayreuth.de