Representing Text (TF-IDF, Word2vec)

Zdeněk Kasner

reusing materials by Jindřich Libovický and Milan Straka

November 11, 2025

Charles University
University

Charles University in Prague
Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics

Today's Lecture Objectives

After this lecture you should be able to

- Use TF-IDF for representing documents and explain its information-theoretical interpretation.
- Explain training of Word2Vec as a special case of logistic regression.
- Use pre-trained word embeddings for simple NLP tasks.

Evaluation in Natural Language Processing

Metrics for Exemplary NLP Tasks

Part-of-speech tagging: assign a part-of-speech to every word in the input text.

Exactly one class is predicted for every word.

Accuracy is the same as micro-averaged precision, recall, and F_1-score, because TP+FP = TP+FN.

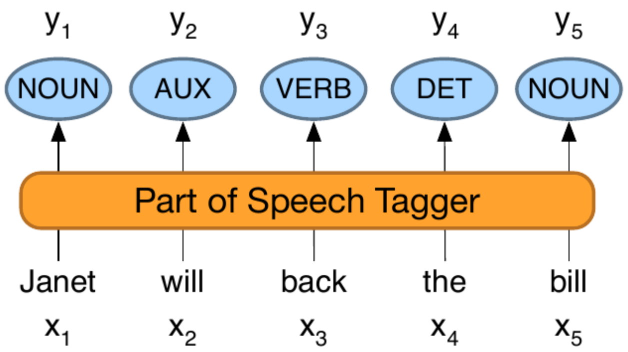


Figure 8.3 of Speech and Language Processing 3rd ed., Jurafsky and Martin, (2024)

Named entity recognition: recognize personal names, organizations, and locations in the input text.

Many words are not a named entity → *accuracy* is artificially high.

Micro-averaged F_1 considers all named entities: "how good are we at recognizing all present named entities".

Macro-averaged F_1: "how good are we at recognizing all named entities **types**".

Words	IO Label	BIO Label
Jane	I-PER	B-PER
Villanueva	I-PER	I-PER
of	O	O
United	I-ORG	B-ORG
Airlines	I-ORG	I-ORG
Holding	I-ORG	I-ORG
discussed	O	O
the	O	O
Chicago	I-LOC	B-LOC
route	O	O
.	O	O

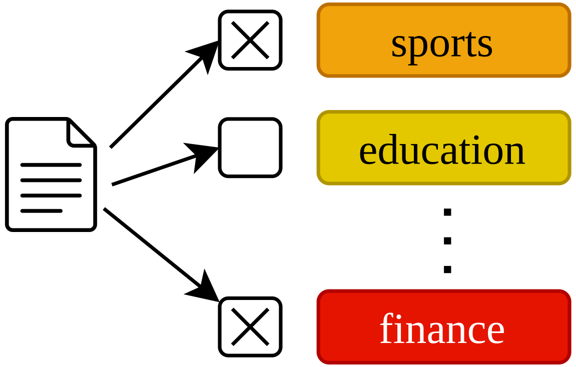
Figure 8.7 of Speech and Language Processing 3rd ed., Jurafsky and Martin, (2024)

Document classification: assign the document to relevant categories (topics).

An input example can be categorized into multiple topics $\rightarrow$ multi-label classification.

Accuracy is very strict (all predicted classes must be exactly the same).

Commonly evaluated using micro-averaged or macro-averaged F_1-score.



TF-IDF

Document Representation

We already know how to represent images and categorical variables (classes, letters, words, …).

Now consider the problem of representing a whole *document*.

An elementary approach is to represent a document as a **bag of words** – we create a feature space with a dimension for every unique word (or for character sequences), called a **term**.

However, there are many ways in which the values of the terms can be set.

Commonly used ways of setting the term values:

- **binary indicators**: 1/0 depending on whether a term is present in a document or not;
- **term frequency (TF)**: relative frequency of a term in a document;

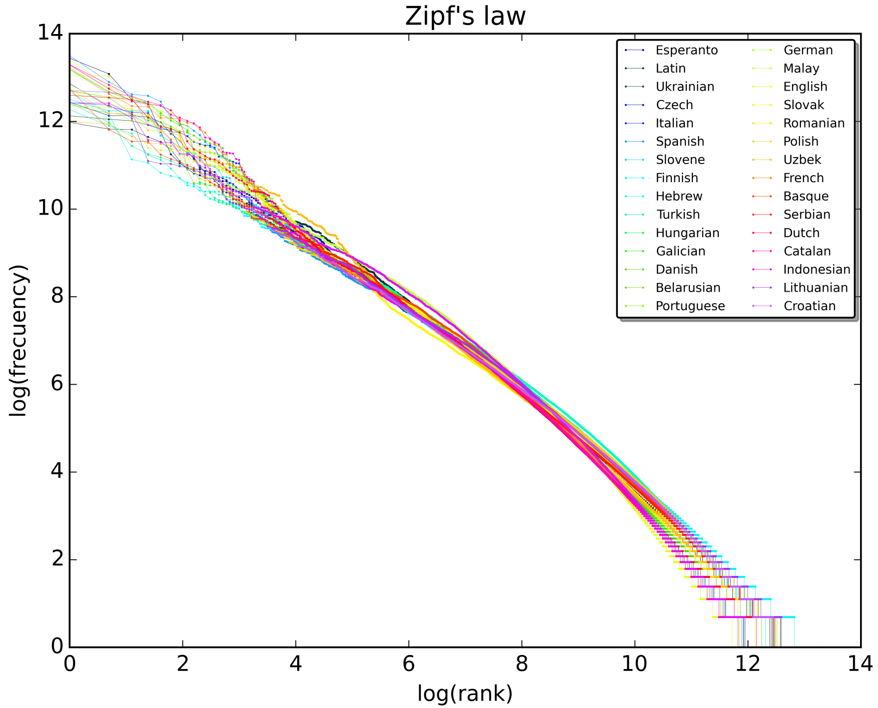
$$TF(t; d) = \frac{\text{number of occurrences of } t \text{ in the document } d}{\text{number of terms in the document } d}$$

- **inverse document frequency (IDF)**: we could also represent a term using self-information of a probability of a random document containing it (therefore, terms with lower document probability have higher weights);

$$IDF(t) = \log \frac{\text{number of documents}}{\text{number of documents containing } t \ \big(\text{optionally} + 1\big)} = I\big(P(d \ni t)\big)$$

- **TF-IDF**: empirically, product $TF \cdot IDF$ reflects quite well how important a term is to a document in a corpus (used by the majority of text-based recommender systems in 2010s).

Original Motivation



https://en.wikipedia.org/wiki/Tf%E2%80%93idf

Jones (1972) provided only intuitive justification.

Zipf's law: empirically, word frequencies follow

$$\text{word frequency} \propto \frac{1}{\text{word rank}}$$

I.e., $\frac{|\mathcal{D}|}{|\{d \in \mathcal{D} : t \in d\}|}$ would be extremely low for frequent words, and high for infrequent ones.

Logarithm normalizes that.

Mutual Information

Mutual Information

Consider two random variables x and y with distributions $\mathrm{x} \sim X$ and $\mathrm{y} \sim Y$.

The conditional entropy $H(Y|X)$ can be naturally considered an expectation of a self-information of $Y|X$, so in the discrete case,

$$H(Y|X) = \mathbb{E}_{\mathrm{x},\mathrm{y}}\big[I(y|x)\big] = -\sum_{x,y} P(x,y) \log P(y|x).$$

In order to assess the amount of information *shared* between the two random variables, we might consider the difference

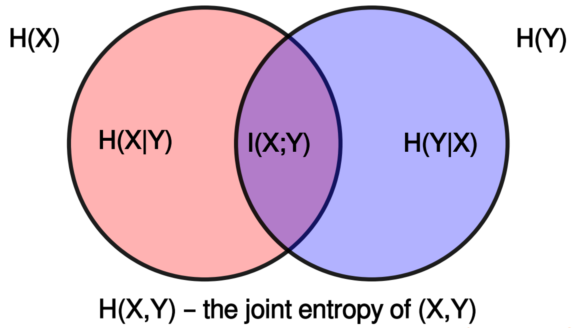
$$\color{red}{H(Y)} - \color{blue}{H(Y|X)} = \mathbb{E}_{\mathrm{x},\mathrm{y}}\big[\color{red}{-\log P(y)}\big] - \mathbb{E}_{\mathrm{x},\mathrm{y}}\big[\color{blue}{-\log P(y|x)}\big] = \mathbb{E}_{\mathrm{x},\mathrm{y}}\left[\log \frac{\color{blue}{P(x,y)}}{\color{blue}{P(x)}\color{red}{P(y)}}\right].$$

We can interpret this value as

> *How many bits of information will we learn about* Y *when we find out* X*?*

Let us denote this quantity as the **mutual information** $I(X;Y)$:

$$I(X;Y) = \mathbb{E}_{\mathrm{x,y}}\left[\log \frac{P(x,y)}{P(x)P(y)}\right].$$



Modification of https://commons.wikimedia.org/wiki/File:Entropy-mutual-information-relative-entropy-relation-diagram.svg

- The mutual information is symmetrical, so

$$I(X;Y) = I(Y;X) = H(Y) - H(Y|X) = H(X) - H(X|Y).$$

- It is easy to verify that

$$I(X;Y) = D_{\mathrm{KL}}\big(P(X,Y) \| P(X)P(Y)\big).$$

  Therefore,
  - $I(X;Y) \geq 0$,
  - $I(X;Y) = 0$ iff $P(X,Y) = P(X)P(Y)$ iff the random variables are independent.

TF-IDF as Mutual Information

Let $\mathcal{D}$ be a collection of documents and $\mathcal{T}$ a collection of terms.

We assume that whenever we need to draw a document, we do it uniformly randomly. Then,

- $P(d) = 1/|\mathcal{D}|$ and $I(d) = H(\mathcal{D}) = \log|\mathcal{D}|$,
- $P(d|t \in d) = 1/|\{d \in \mathcal{D} : t \in d\}|$,
- $I(d|t \in d) = H(\mathcal{D}|t) = \log|\{d \in \mathcal{D} : t \in d\}|$, assuming $0 \cdot \log 0 = 0$ in H as usual,
- $I(d) - I(d|t \in d) = H(\mathcal{D}) - H(\mathcal{D}|t) = \log \frac{|\mathcal{D}|}{|\{d \in \mathcal{D} : t \in d\}|} = \mathit{IDF}(t).$

Finally, we can compute the mutual information $I(\mathcal{D}; \mathcal{T})$ as

$$I(\mathcal{D}; \mathcal{T}) = \sum_{d,\, t \in d} P(d) \cdot P(t|d) \cdot \big(I(d) - I(d|t)\big) = \frac{1}{|\mathcal{D}|} \sum_{d,\, t \in d} \mathit{TF}(t; d) \cdot \mathit{IDF}(t).$$

Therefore, summing all TF-IDF terms recovers the mutual information between $\mathcal{D}$ and $\mathcal{T}$, and we can say that each TF-IDF carries a “bit of information” attached to a document-term pair.

Word2Vec

Representation Learning

We interpreted MLP as automatic feature extraction for a generalized linear model.

Representation learning: learning using a proxy task that leads to reusable features.

Famous example: pre-training image representations using **object classification**:

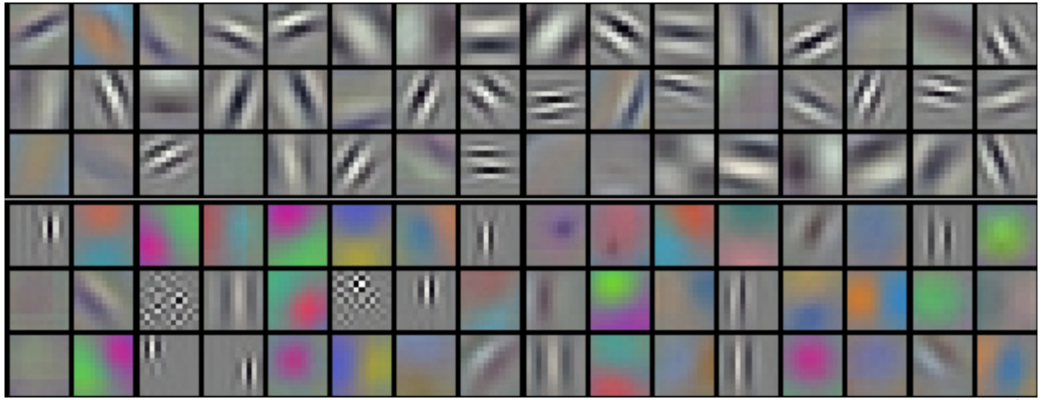


Figure 3 of ImageNet Classification with Deep Convolutional Neural Networks, Krizhevsky et al. (2012)

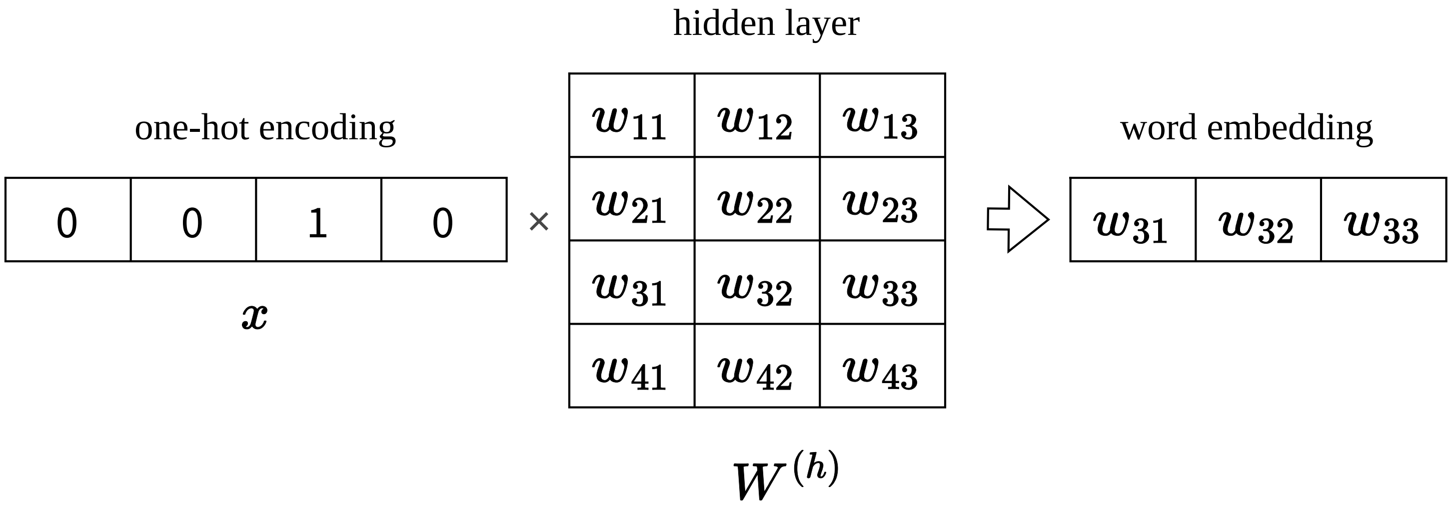
Can we learn such features for **representing text**?

Word Embeddings

We represent an input word with a one-hot vector (assuming a limited vocabulary).

Multiplying the one-hot vector with a weight matrix = picking a row from the weight matrix.

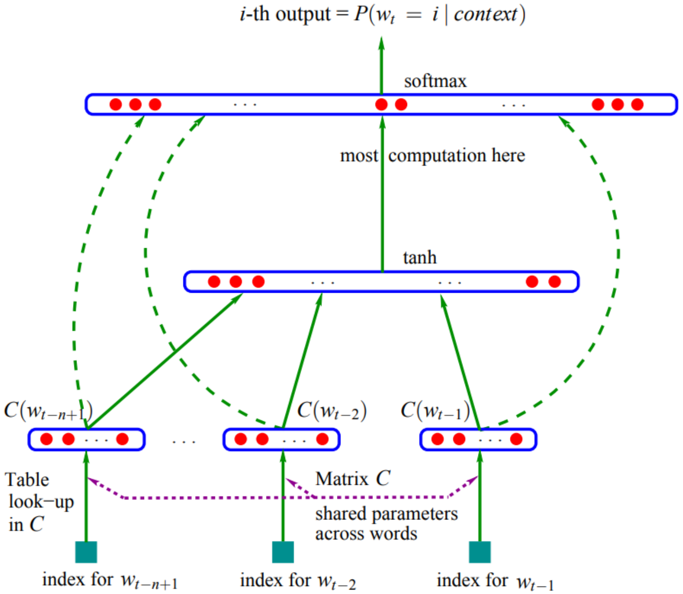
We call this row a **word embedding**.



Origins of Word Embeddings

In 2003, Bengio et al. used MLP for language modeling: predicting a probability of the next word.

They reused the embedding matrix for all input words (regardless of their position).



Bengio, Yoshua, Réjean Ducharme, and Pascal Vincent. "A neural probabilistic language model." Advances in neural information processing systems 13 (2000). Figure 1.

Origins of Word Embeddings

Collobert et al. (2011) first reused word embeddings as features for other NLP tasks.

Geometric properties: neighbors in the embedding space are semantically similar words.

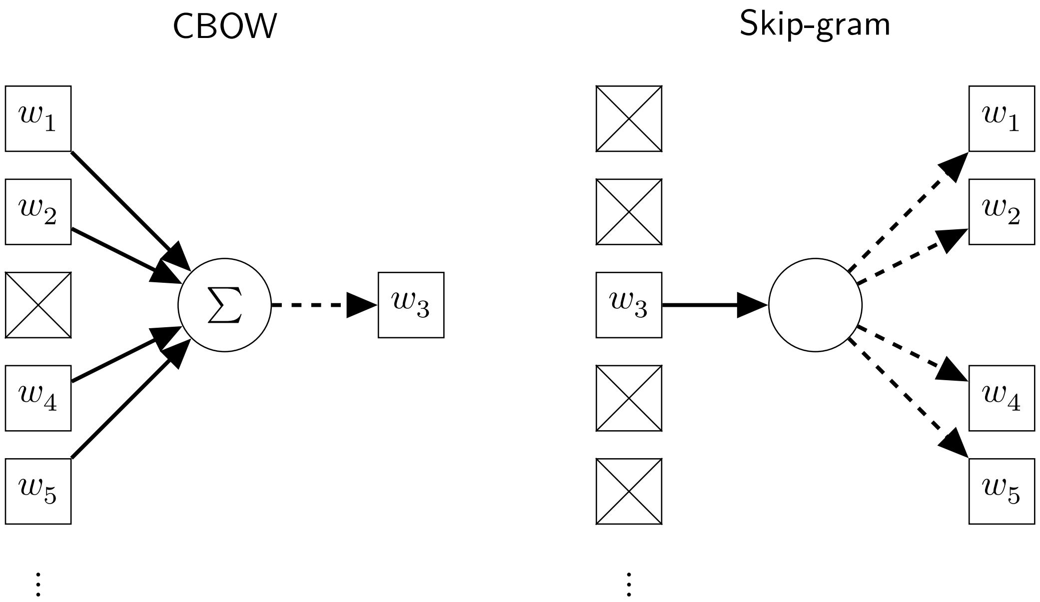
FRANCE 454	JESUS 1973	XBOX 6909	REDDISH 11724	SCRATCHED 29869	MEGABITS 87025
AUSTRIA	GOD	AMIGA	GREENISH	NAILED	OCTETS
BELGIUM	SATI	PLAYSTATION	BLUISH	SMASHED	MB/S
GERMANY	CHRIST	MSX	PINKISH	PUNCHED	BIT/S
ITALY	SATAN	IPOD	PURPLISH	POPPED	BAUD
GREECE	KALI	SEGA	BROWNISH	CRIMPED	CARATS
SWEDEN	INDRA	PSNUMBER	GREYISH	SCRAPED	KBIT/S
NORWAY	VISHNU	HD	GRAYISH	SCREWED	MEGAHERTZ
EUROPE	ANANDA	DREAMCAST	WHITISH	SECTIONED	MEGAPIXELS
HUNGARY	PARVATI	GEFORCE	SILVERY	SLASHED	GBIT/S
SWITZERLAND	GRACE	CAPCOM	YELLOWISH	RIPPED	AMPERES

Collobert, Ronan, et al. "Natural language processing (almost) from scratch." Journal of machine learning research 12.(2011): 2493-2537. Table 6.

In a downstream task, we can learn something also for words that **were not in training data** but are similar to some that were.

How to get the word embeddings without training a computationally expensive model?

1. Simplify the **input context**: treat it as a bag of words.
2. Simplify the **model architecture**: turn it to a linear model.



Word2Vec: Skip-gram Sampling

1.	**All** human beings	are born free and equal in dignity ...	→	(All, humans) (All, beings)
2.	All **human** beings are	born free and equal in dignity ...	→	(human, All) (human, beings) (human, are)
3.	All human **beings** are born	free and equal in dignity ...	→	(beings, All) (beings, human) (beings, are) (beings, born)
4.	All [human beings **are** born free]	and equal in dignity ...	→	(are, human) (are, beings) (are, born) (are, free)

SkipGram: Basic Formulation

For each word w from the vocabulary V, we want to learn a d-dimensional embedding vector $\boldsymbol{e}_w \in \mathbb{R}^d$.

We train a feed-forward model with two layers:

- The first layer is the input embedding matrix $\boldsymbol{E} \in \mathbb{R}^{|V| \times d}$ (without any non-linearity).
- The second layer is the output matrix $\boldsymbol{W} \in \mathbb{R}^{d \times |V|}$ followed by the $\operatorname{softmax}$ activation function.

The model computes the probability of words $c \in V$ appearing in the context of w.

After training, we use the rows of the embedding matrix $\boldsymbol{E}$ as word embeddings (in Word2Vec, the output matrix gets discarded).

SkipGram: Towards Better Efficiency

The vocabulary V contains $\sim 10^5 - 10^6$ word forms $\Rightarrow$ computing the $\operatorname{softmax}$ would be expensive.

To solve this, we turn the problem into **binary classification**: we predict the probability for each pair of words independently using logistic regression.

For the input word w with an embedding $\boldsymbol{e}_w \in \boldsymbol{E}$ and the context word c_i with an output embedding $\boldsymbol{v}_{c_i} \in \boldsymbol{W}$, we compute the probability of their co-occurence as:

$$P(c|w) = \sigma(\boldsymbol{e}_w^T \boldsymbol{v}_c).$$

We compute the loss as $-\log \sigma(\boldsymbol{e}_w^T \boldsymbol{v}_{c_i})$.

More generally, we say that $\sigma\left(\boldsymbol{E}\boldsymbol{W}\right)_{i,j}$ estimates a table of $|V| \times |V|$ with probabilities that j-th word is in the neighbor window of i-th word.

SkipGram: Negative Sampling

In the previous formulation, our model was given only positive examples: each pair of words w and c **do** occur in the same context.

To present the model with negative examples, we also sample K words c_j that are **not** in the context window.

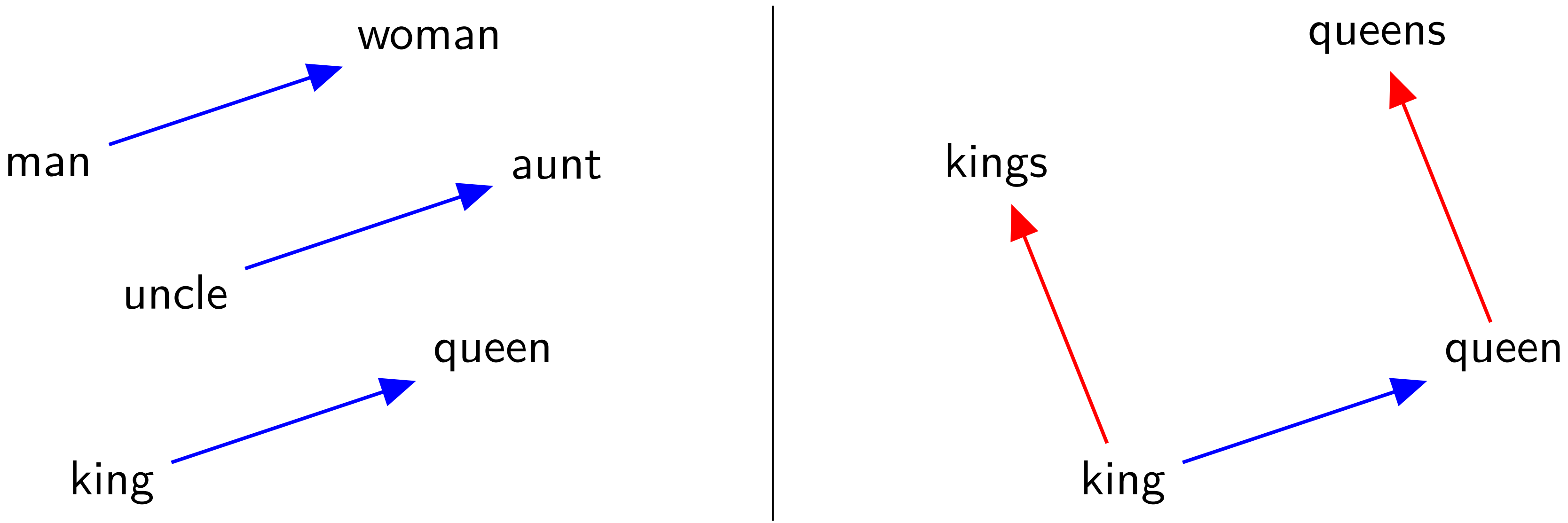
These words contribute to the loss function:

$$L = -\log \sigma(\boldsymbol{e}_w^T \boldsymbol{v}_{c_i}) - \sum_{j=1}^{K} \log(1 - \sigma(\boldsymbol{e}_w^T \boldsymbol{v}_{c_j})).$$

The usual value of negative samples is $K = 5$, but it can be even $K = 2$ for extremely large corpora.

Interesting Properties of Word2Vec

Vector arithmetics seems to capture lexical semantics.



Mikolov, Tomáš, Wen-tau Yih, and Geoffrey Zweig. "Linguistic regularities in continuous space word representations." Proceedings of NAACL-HLT. 2013. Adapted from Figure 2.

Using Word Embeddings

We can use word embeddings for downstream NLP tasks.

Text classification

- Tasks: topic classification, sentiment analysis, natural language inference.
- Problem: we need a fixed-length representation for a text variable length.
- Solution: compute an average or sum over the sequence of embeddings.

Token classification (also called sequence labeling):

- Tasks: POS tagging, named entity recognition, extractive summarization.
- Problem: we would like to integrate a context for each word.
- Solution: use a sliding window over embeddings and classify the middle one.

More Recent Word Embeddings

- GloVe (Global Vectors): Takes into account global co-occurences of words.
- FastText: Word embedding is a sum of substring embeddings $\rightarrow$ can generate embeddings of unseen words.
- Backpack Language Models (Hewitt et al., 2023): combining multiple sense vectors for each word.
- State of the art: contextual embeddings (BERT), large language models (LLM2Vec).

Today's Lecture Objectives

After this lecture you should be able to:

- Use TF-IDF for representing documents and explain its information-theoretical interpretation.
- Explain training of Word2Vec as a special case of logistic regression.
- Use pre-trained word embeddings for simple NLP tasks.