Used to represent a repertoire of characters by an encoding system that assigns a number to each character for digital representation, e.g. UTF-8, Unicode, …

https://en.wikipedia.org/wiki/Harvard_Mark_I https://en.wikipedia.org/wiki/Character_encoding

Used to represent a repertoire of characters by an encoding system that assigns a number to each character for digital representation, e.g. UTF-8, Unicode, …

Harvard Mark I
1944, 3.7kw, 4.3t

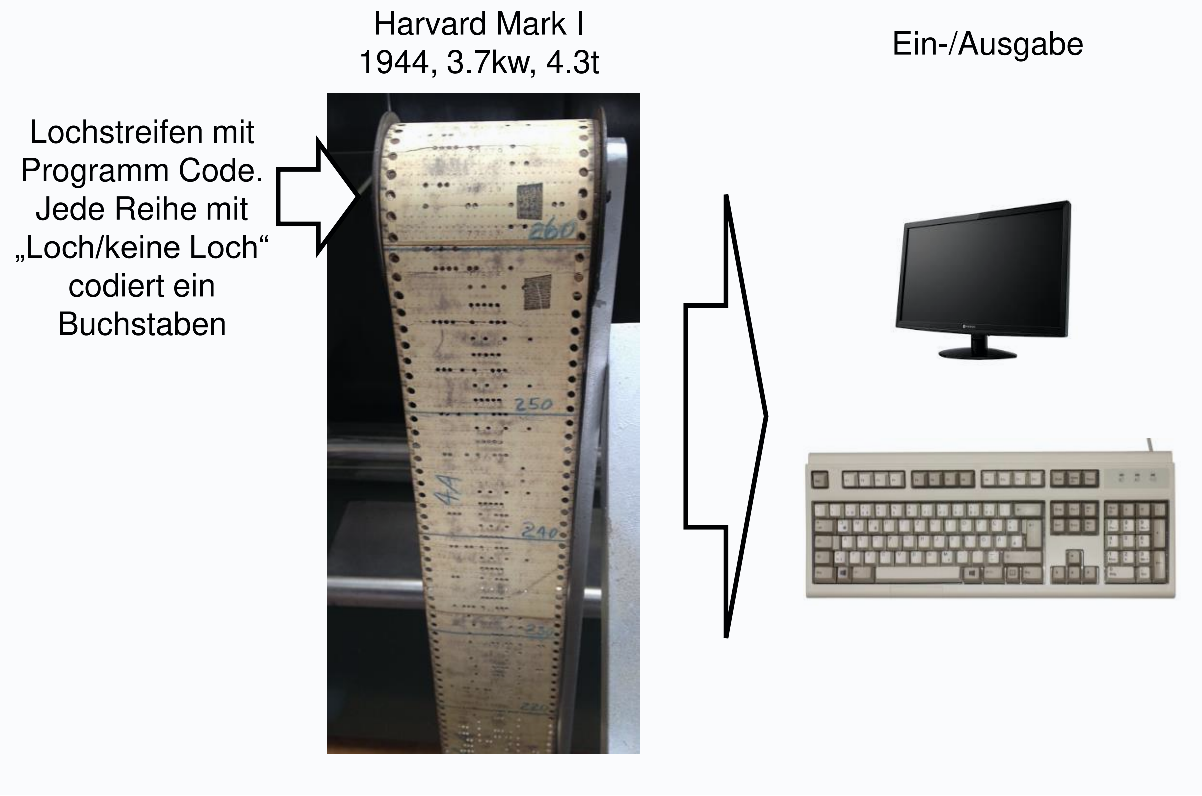
Lochstreifen mit Programm Code. Jede Reihe mit „Loch/keine Loch“ codiert ein Buchstaben

https://en.wikipedia.org/wiki/Harvard_Mark_I https://en.wikipedia.org/wiki/Character_encoding

Word Embedding

From Punched Tape to Character encoding and Text-Preprocessing

Used to represent a repertoire of characters by an encoding system that assigns a number to each character for digital representation, e.g. UTF-8, Unicode, …



https://en.wikipedia.org/wiki/Harvard_Mark_I https://en.wikipedia.org/wiki/Character_encoding

Word Embedding

From Punched Tape to Character encoding and Text-Preprocessing

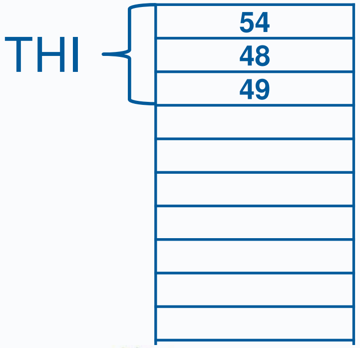
Used to represent a repertoire of characters by an encoding system that assigns a number to each character for digital representation, e.g. UTF-8, Unicode, …

Harvard Mark I
1944, 3.7kw, 4.3t

Lochstreifen mit
Programm Code.
Jede Reihe mit
„Loch/keine Loch“
codiert ein
Buchstaben

Ein-/Ausgabe

Computer
(UTF8 Encoding)



SSD >1TB (2020)

https://en.wikipedia.org/wiki/Harvard_Mark_I https://en.wikipedia.org/wiki/Character_encoding

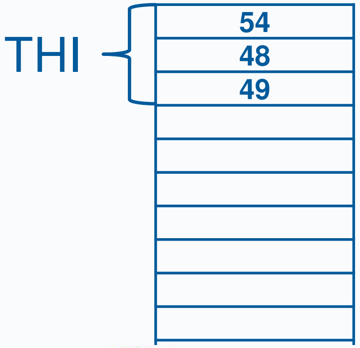
Used to represent a repertoire of characters by an encoding system that assigns a number to each character for digital representation, e.g. UTF-8, Unicode, …

Harvard Mark I
1944, 3.7kw, 4.3t

Lochstreifen mit Programm Code. Jede Reihe mit „Loch/keine Loch“ codiert ein Buchstaben

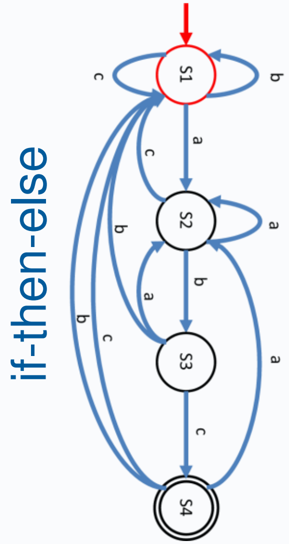
Ein-/Ausgabe

Computer
(UTF8 Encoding)



SSD >1TB (2020)

Textvorverarbeitung

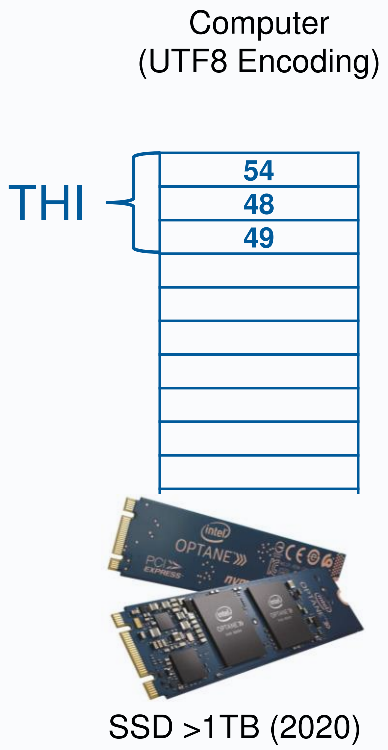


https://en.wikipedia.org/wiki/Harvard_Mark_I https://en.wikipedia.org/wiki/Character_encoding

Mathematical view of a Character Encoding

The data is encoded by a sequence of numbers. Let’s try to interpret these numbers, mathematically.

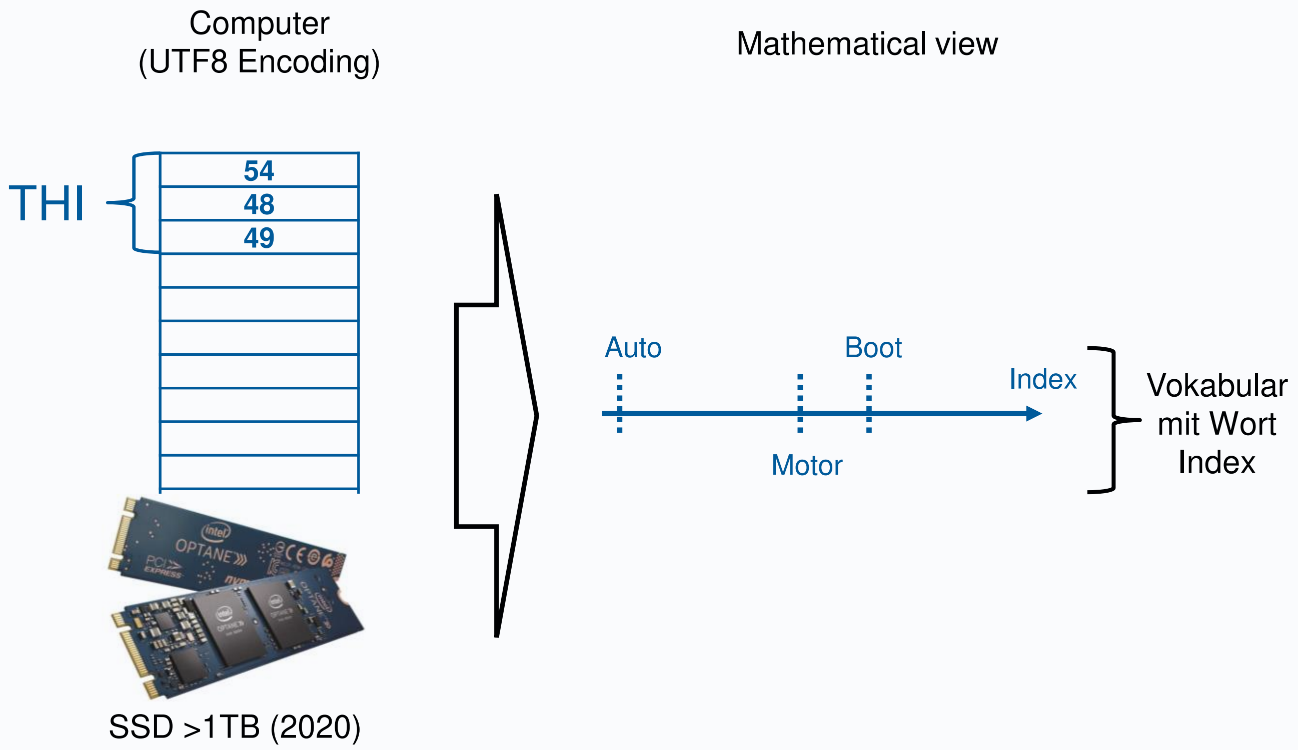
The data is encoded by a sequence of numbers. Let's try to interpret these numbers, mathematically.



Word Embedding

Mathematical view of a Character Encoding

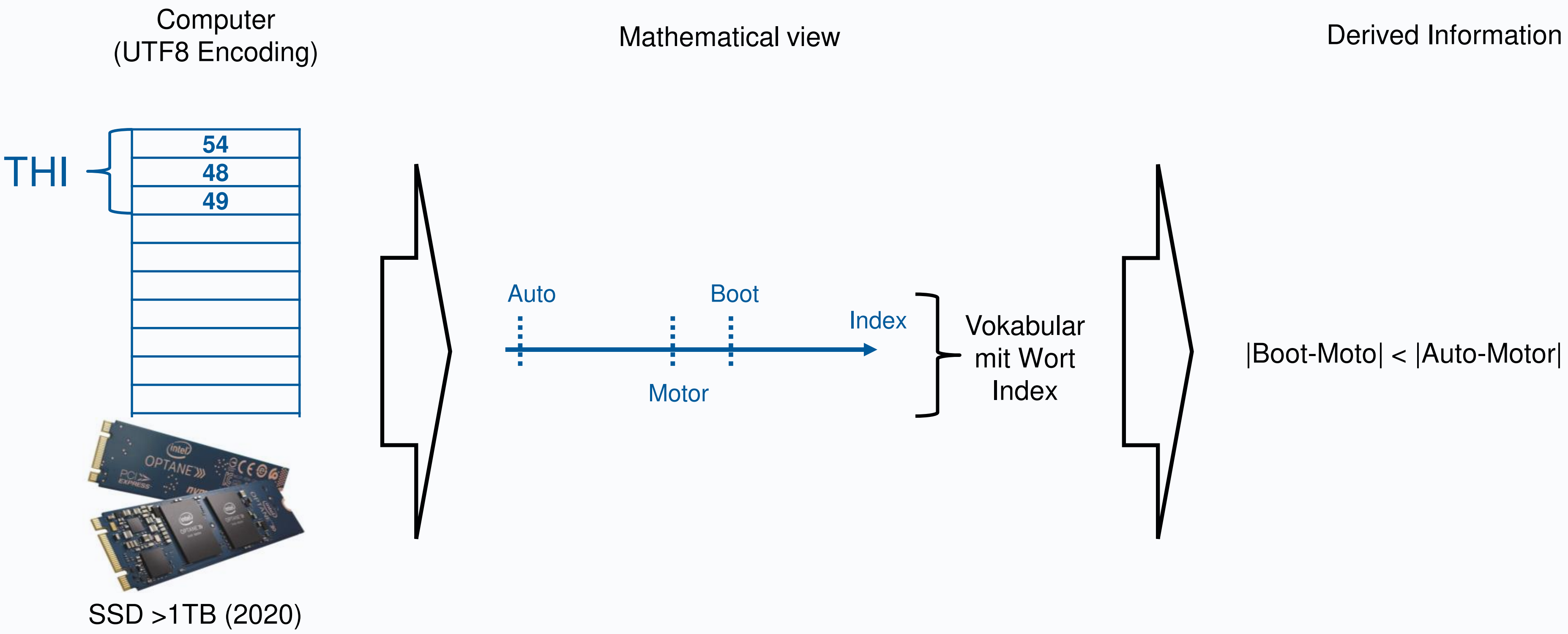
The data is encoded by a sequence of numbers. Let’s try to interpret these numbers, mathematically.



Word Embedding

Mathematical view of a Character Encoding

The data is encoded by a sequence of numbers. Let’s try to interpret these numbers, mathematically.



Word Embedding

Mathematical view of a Character Encoding

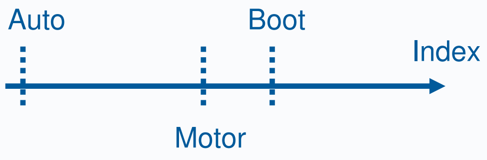
The data is encoded by a sequence of numbers. Let's try to interpret these numbers, mathematically.

Computer (UTF8 Encoding)	Mathematical view	Derived Information	Derived Knowledge

THI

54
48
49

SSD >1TB (2020)



Vokabular mit Wort Index

|Boot-Moto| < |Auto-Motor|

Ein Boot hat ein Motor, ein Auto hat kein Motor

Word Embedding

Mathematical view of a Character Encoding

The data is encoded by a sequence of numbers. Let’s try to interpret these numbers, mathematically.

Computer
(UTF8 Encoding)

THI

54
48
49

SSD >1TB (2020)

Mathematical view

Auto
Boot
Index
Motor

Vokabular mit Wort Index

Derived Information

|Boot-Moto| < |Auto-Motor|

Derived Knowledge

Ein Boot hat ein Motor,
ein Auto hat kein Motor

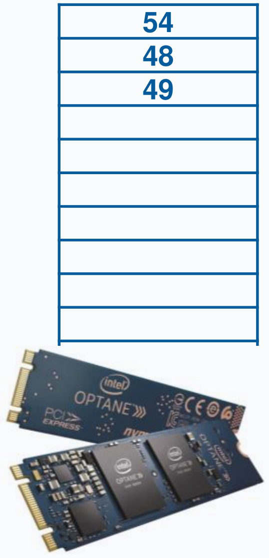
Doesn‘t work. No meaningful semantics is deducable

Data using a structure and language consistent with first-order predicate logic, first described in 1969 by Edgar F. Codd, where all data is represented in terms of tuples, grouped into relations.

https://en.wikipedia.org/wiki/Harvard_Mark_I https://en.wikipedia.org/wiki/Character_encoding

Data using a structure and language consistent with first-order predicate logic, first described in 1969 by Edgar F. Codd, where all data is represented in terms of tuples, grouped into relations.

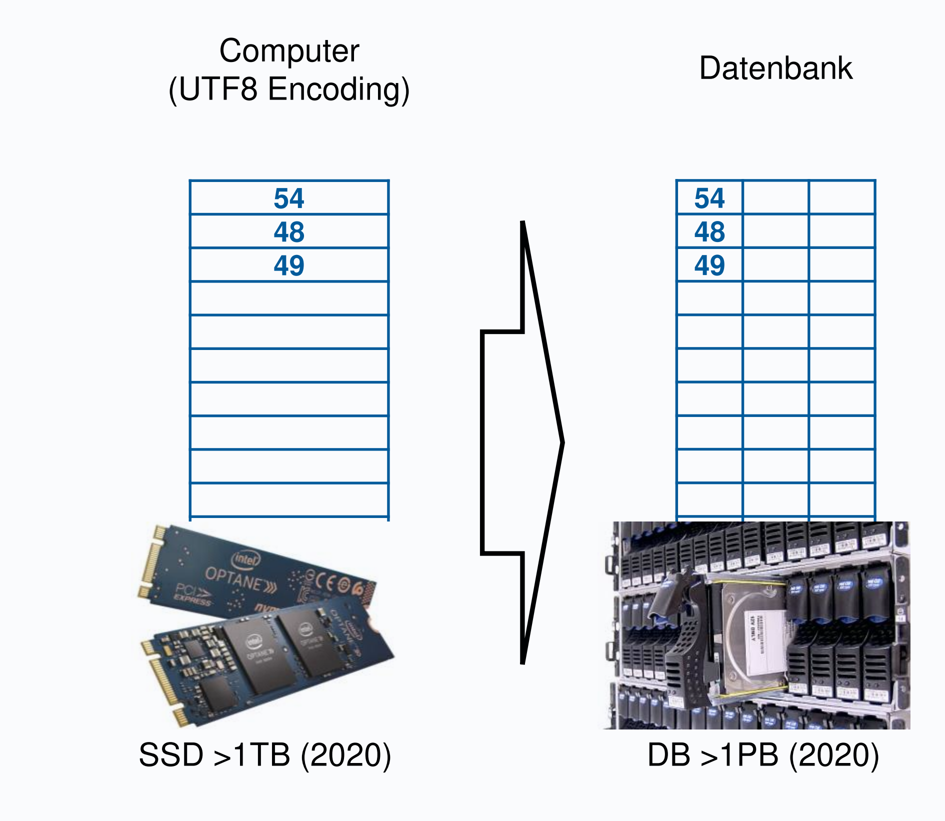
Computer
(UTF8 Encoding)



SSD >1TB (2020)

https://en.wikipedia.org/wiki/Harvard_Mark_I https://en.wikipedia.org/wiki/Character_encoding

Data using a structure and language consistent with first-order predicate logic, first described in 1969 by Edgar F. Codd, where all data is represented in terms of tuples, grouped into relations.

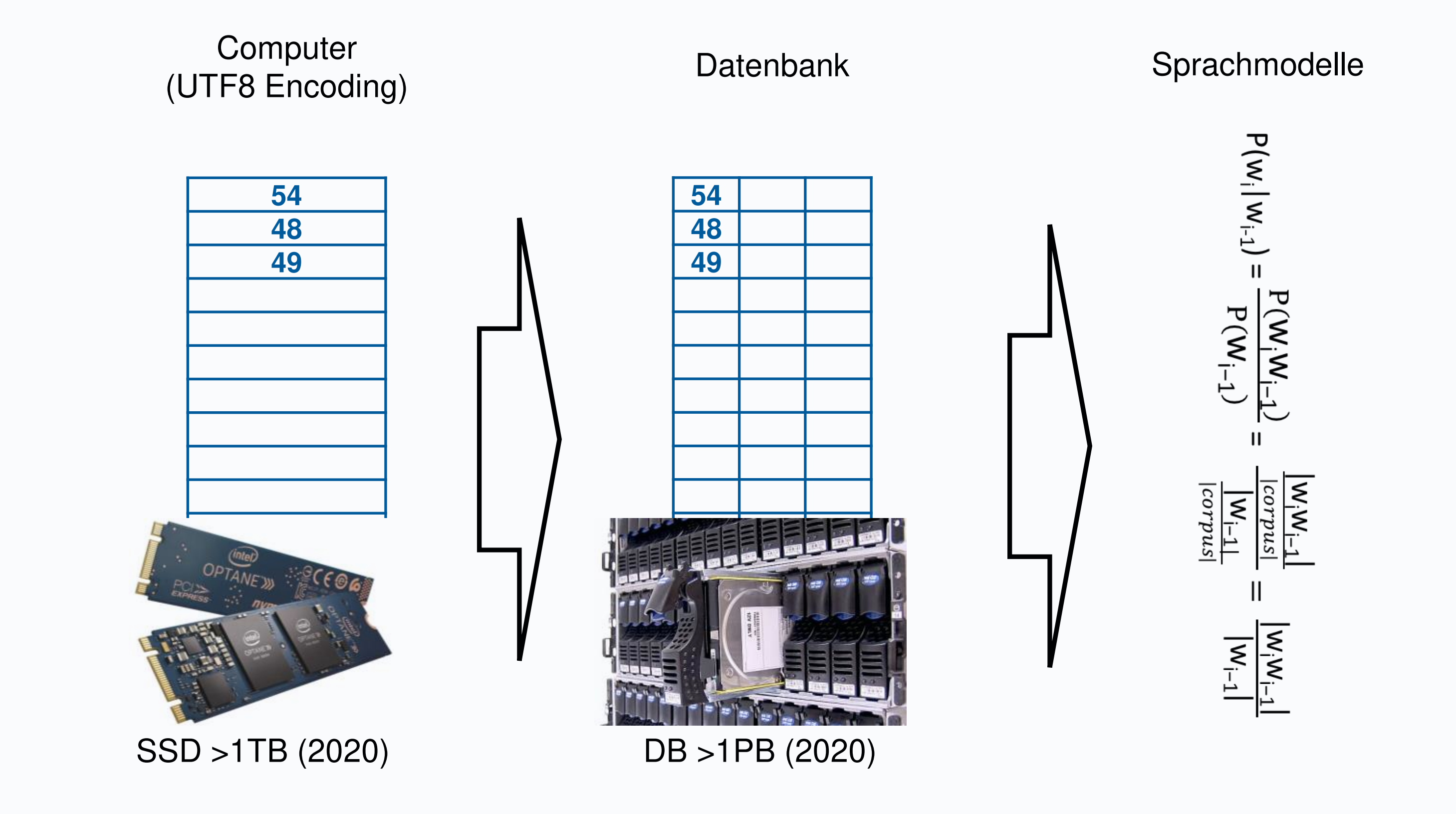


https://en.wikipedia.org/wiki/Harvard_Mark_I https://en.wikipedia.org/wiki/Character_encoding

Word Embedding

From storing characters to storing relational data

Data using a structure and language consistent with first-order predicate logic, first described in 1969 by Edgar F. Codd, where all data is represented in terms of tuples, grouped into relations.



https://en.wikipedia.org/wiki/Harvard_Mark_I https://en.wikipedia.org/wiki/Character_encoding

Word Embedding

From storing characters to storing relational data

Data using a structure and language consistent with first-order predicate logic, first described in 1969 by Edgar F. Codd, where all data is represented in terms of tuples, grouped into relations.

Computer (UTF8 Encoding)

54
48
49

SSD >1TB (2020)

Datenbank

54		
48		
49		

DB >1PB (2020)

Sprachmodelle

$$P(w_i|w_{i-1}) = \frac{P(W_i W_{i-1})}{P(W_{i-1})} = \frac{\frac{|W_i W_{i-1}|}{|corpus|}}{\frac{|W_{i-1}|}{|corpus|}} = \frac{|W_i W_{i-1}|}{|W_{i-1}|}$$

BOW

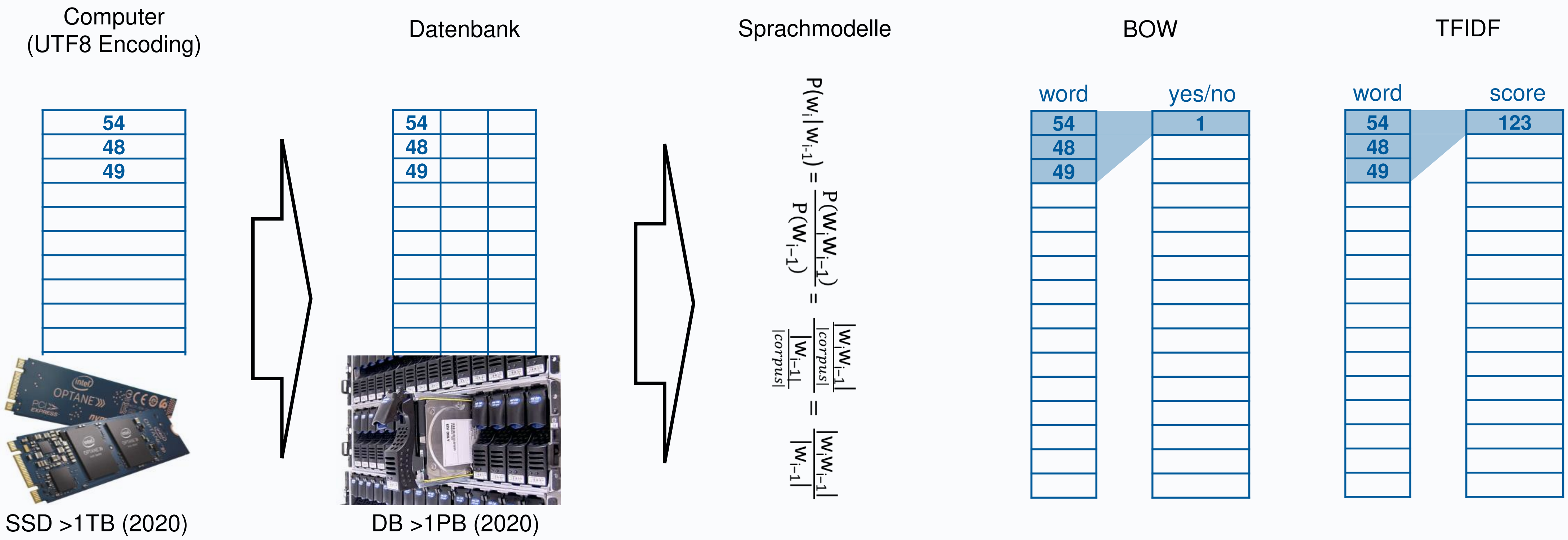
word	yes/no
54 48 49	1

https://en.wikipedia.org/wiki/Harvard_Mark_I https://en.wikipedia.org/wiki/Character_encoding

Word Embedding

From storing characters to storing relational data

Data using a structure and language consistent with first-order predicate logic, first described in 1969 by Edgar F. Codd, where all data is represented in terms of tuples, grouped into relations.



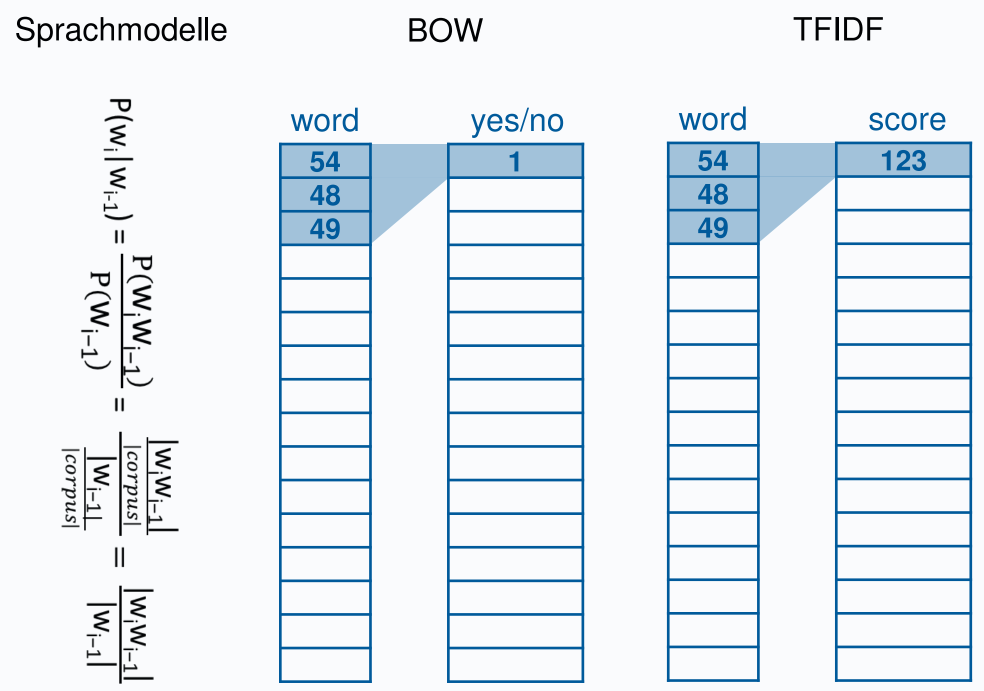
$$P(w_i|w_{i-1}) = \frac{P(W_i W_{i-1})}{P(W_{i-1})} = \frac{\frac{|W_i W_{i-1}|}{|corpus|}}{\frac{|W_{i-1}|}{|corpus|}} = \frac{|W_i W_{i-1}|}{|W_{i-1}|}$$

https://en.wikipedia.org/wiki/Harvard_Mark_I https://en.wikipedia.org/wiki/Character_encoding

Word Embedding

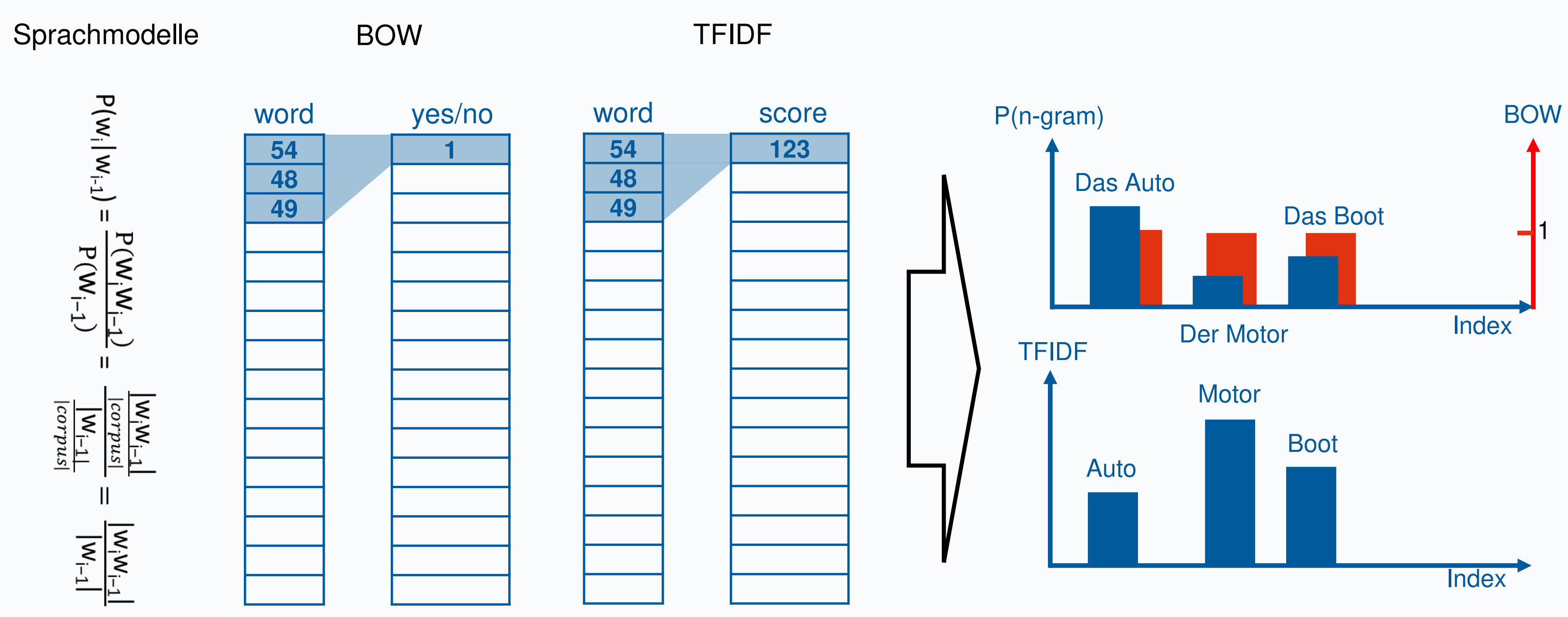
Mathematical view of relational data such as n-gram, BOW, TFIDF

Some information can be derived from n-gram, BOW, TFIDF key-value pairs.



Mathematical view of relational data such as n-gram, BOW, TFIDF

Some information can be derived from n-gram, BOW, TFIDF key-value pairs.



Word Embedding

Mathematical view of relational data such as n-gram, BOW, TFIDF

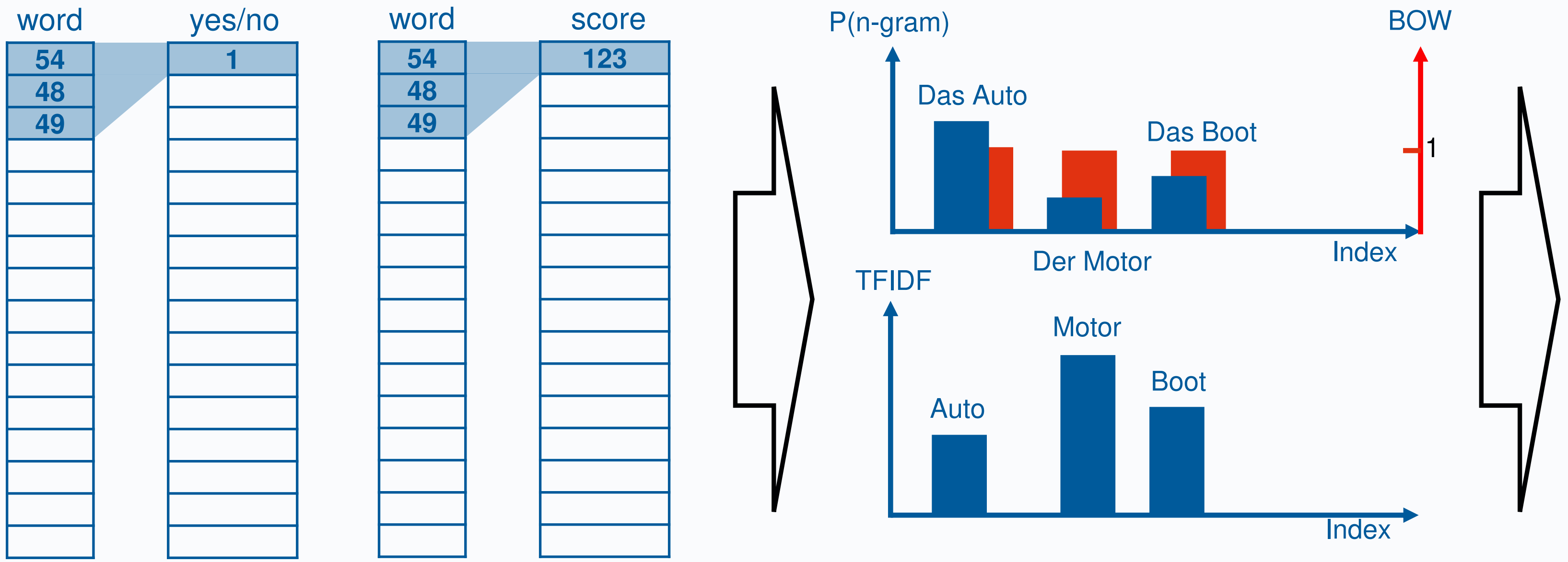
Some information can be derived from n-gram, BOW, TFIDF key-value pairs.

Sprachmodelle

$$P(W_i|W_{i-1}) = \frac{P(W_iW_{i-1})}{P(W_{i-1})} = \frac{\frac{|W_iW_{i-1}|}{|corpus|}}{\frac{|W_{i-1}|}{|corpus|}} = \frac{|W_iW_{i-1}|}{|W_{i-1}|}$$

BOW

TFIDF

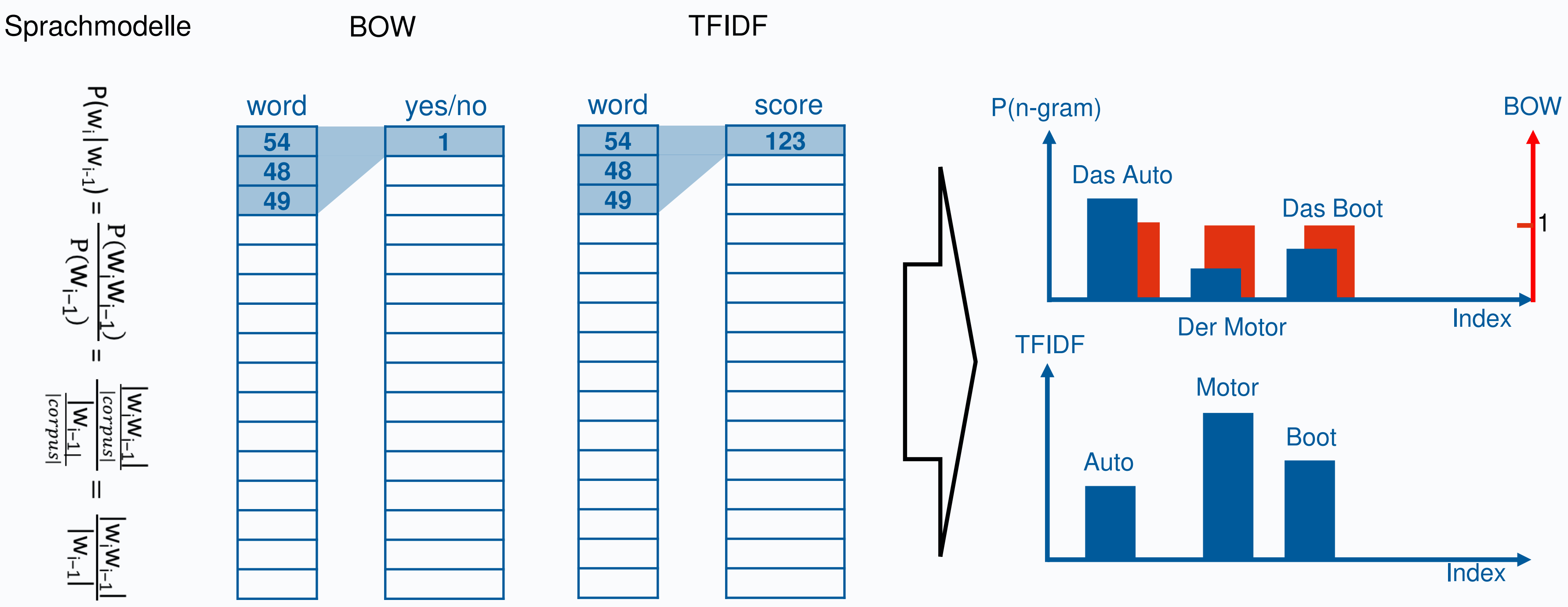


P(Boot) < P(Auto) ?
BOW(Boot) = BOW(Auto)
TFIDF(Boot) > TFIDF(Auto) ?

Word Embedding

Mathematical view of relational data such as n-gram, BOW, TFIDF

Some information can be derived from n-gram, BOW, TFIDF key-value pairs.



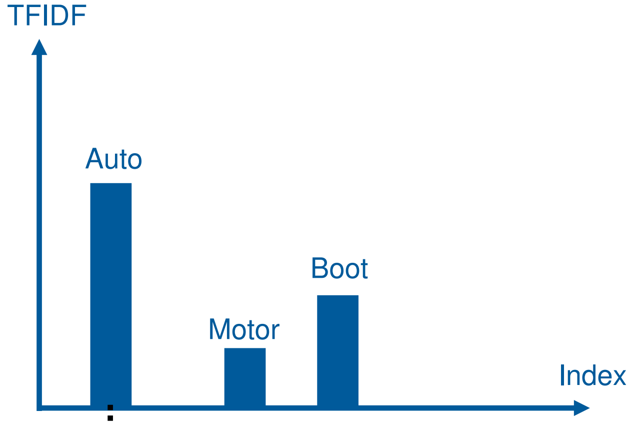
P(Boot) < P(Auto) ?
BOW(Boot) = BOW(Auto)
TFIDF(Boot) > TFIDF(Auto) ?

Bestimmte Semantik ableitbar:
Sprachmodell, Stop Words

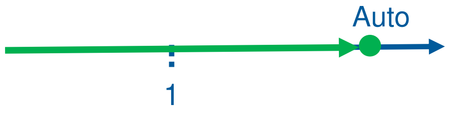
Word Embedding

Word as Vector

Each word is represented by one unique vector.



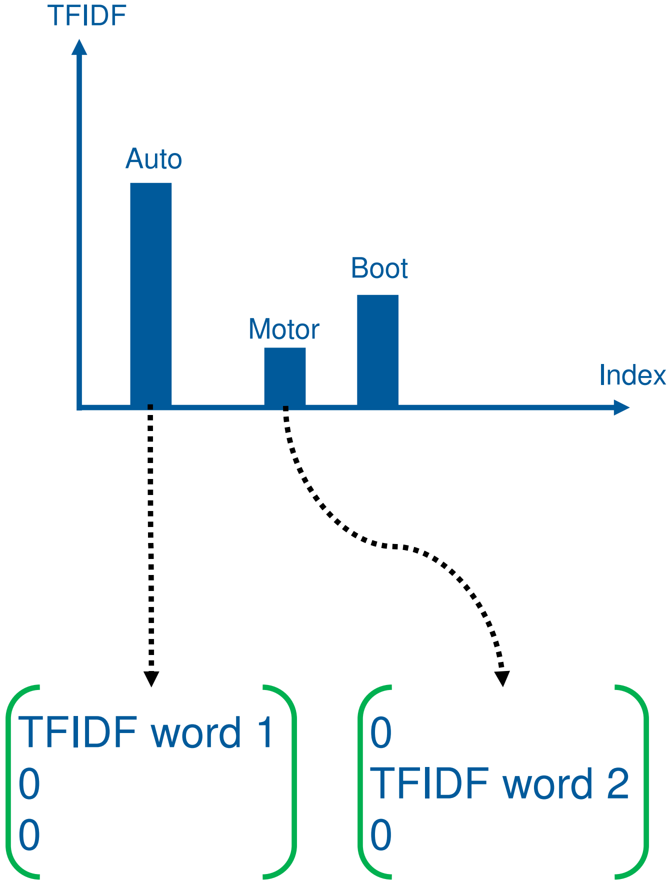
$$\begin{pmatrix} \text{TFIDF word 1} \\ 0 \\ 0 \end{pmatrix}$$

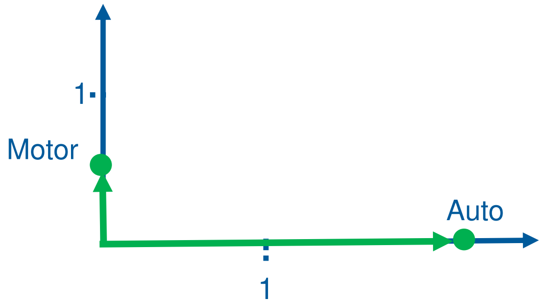


Word Embedding

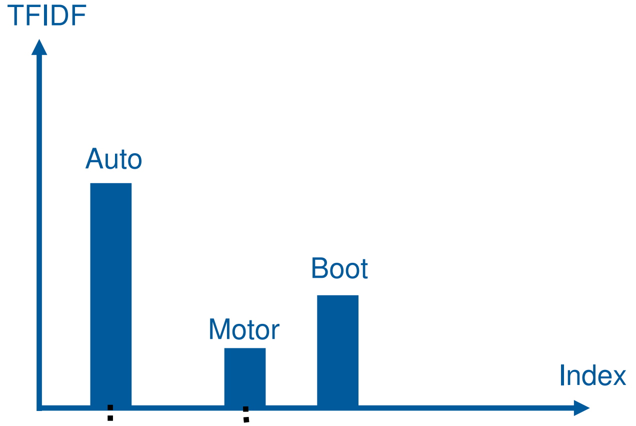
Words as Vectors

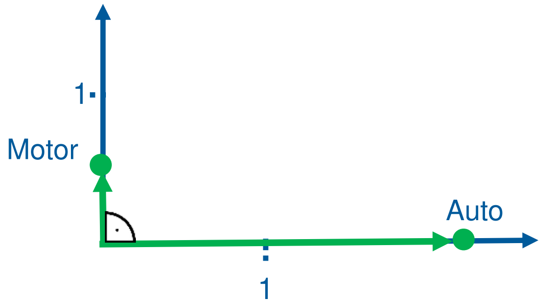
Each word is represented by one unique vector.





Defining word vectors in such a way that there is no semantic meaning between words.





$$\begin{pmatrix} \text{TFIDF word 1} \\ 0 \\ 0 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ \text{TFIDF word 2} \\ 0 \end{pmatrix} = 0$$

Dot-Product (zu Deut. Skalarprodukt):

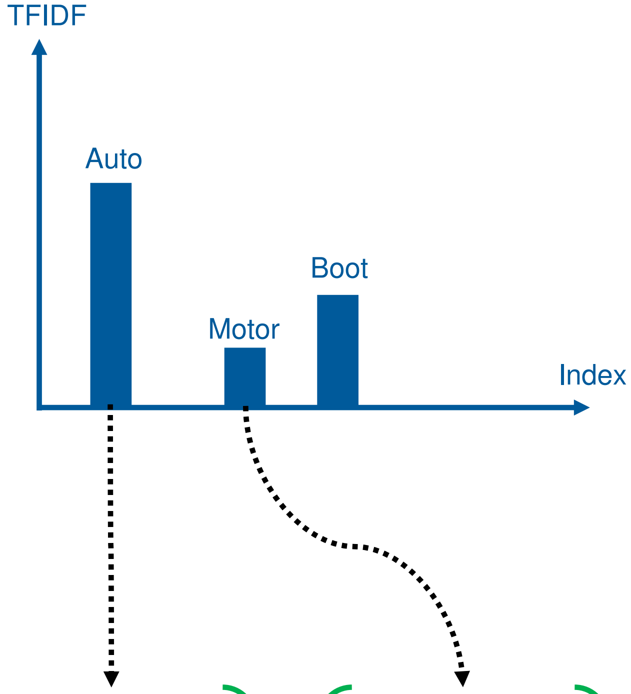
$$\vec{a} \cdot \vec{b} = |\vec{a}|\,|\vec{b}|\,\cos\sphericalangle(\vec{a},\vec{b})$$

$$\vec{a} \perp \vec{b} \iff \vec{a} \cdot \vec{b} = 0$$

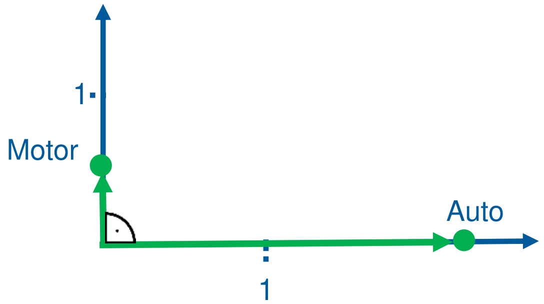
Word Embedding

Words as Vectors

Defining word vectors in such a way that there is no semantic meaning between words.



$$\begin{pmatrix} \text{TFIDF word 1} \\ 0 \\ 0 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ \text{TFIDF word 2} \\ 0 \end{pmatrix} = 0$$



„Motor“ has nothing in common with „Auto“.

Dot-Product:

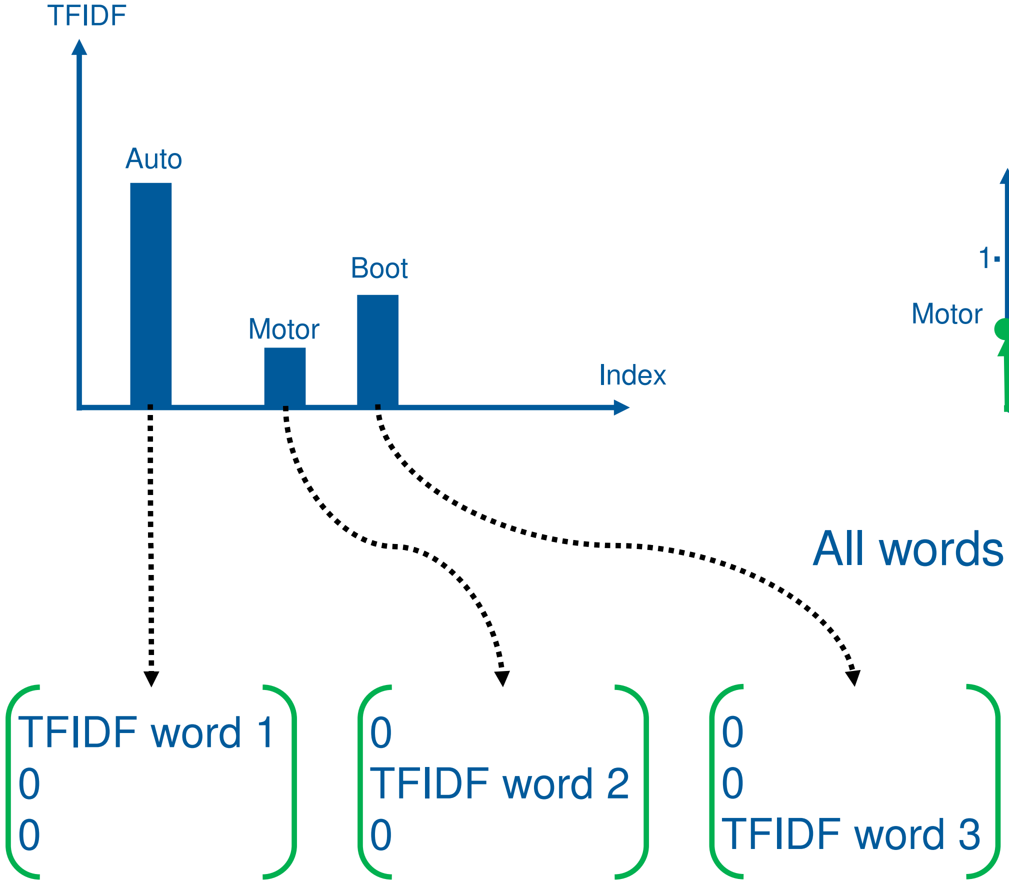
$$\vec{a} \cdot \vec{b} = |\vec{a}|\,|\vec{b}|\, \cos \sphericalangle(\vec{a}, \vec{b})$$

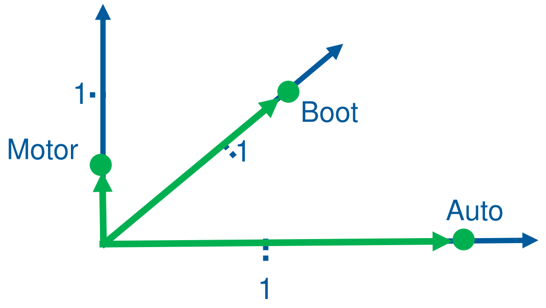
$$\vec{a} \perp \vec{b} \iff \vec{a} \cdot \vec{b} = 0$$

Word Embedding

Words as Vectors

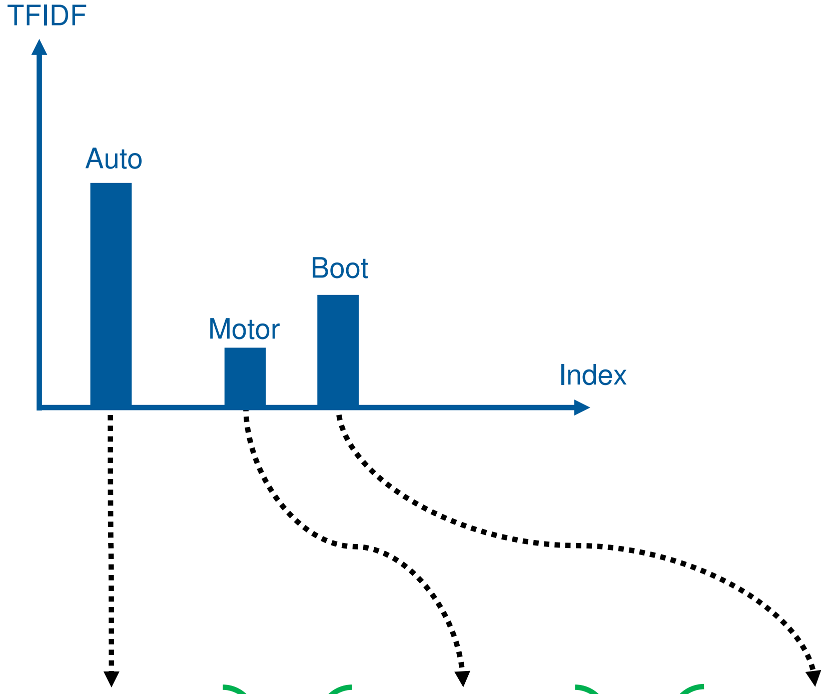
Defining word vectors in such a way that there is no semantic meaning between words.

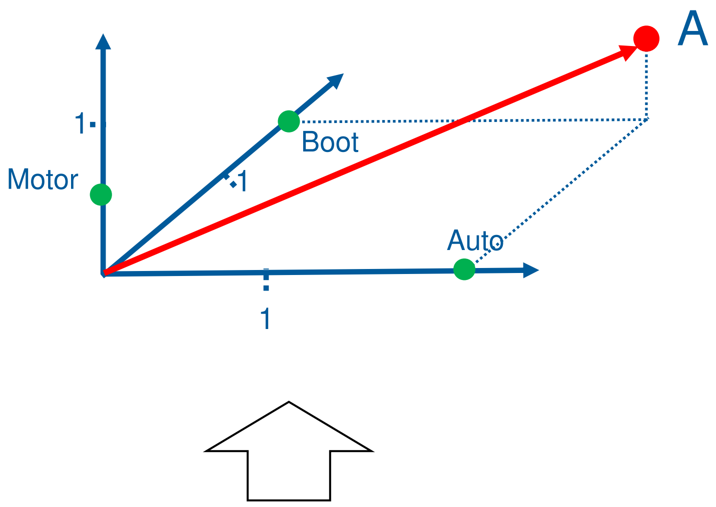




All words independent (orthogonal)

A document (collection of words) is the sum of all word vectors aka “document vector”

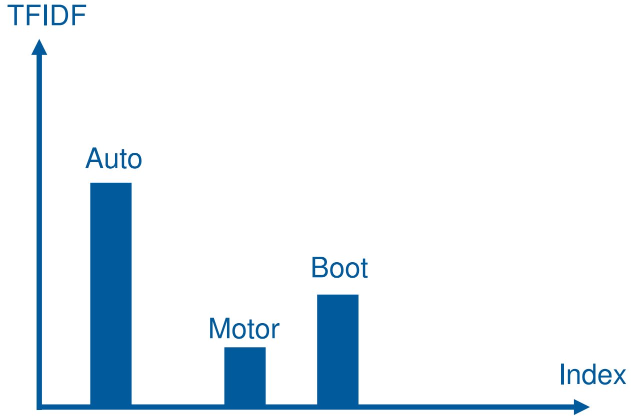


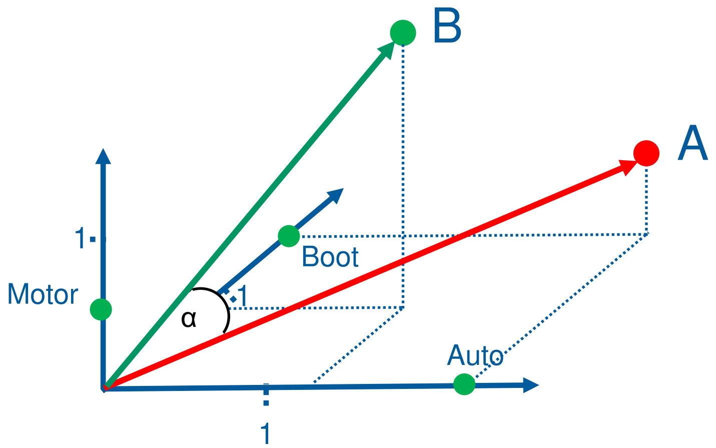


$$\begin{pmatrix} \text{TFIDF word 1} \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ \text{TFIDF word 2} \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ \text{TFIDF word 3} \end{pmatrix} = \begin{pmatrix} \text{TFIDF word 1} \\ \text{TFIDF word 2} \\ \text{TFIDF word 3} \end{pmatrix}$$

Document A

The similarity between documents is measured by the cosine similarity of the document vectors.





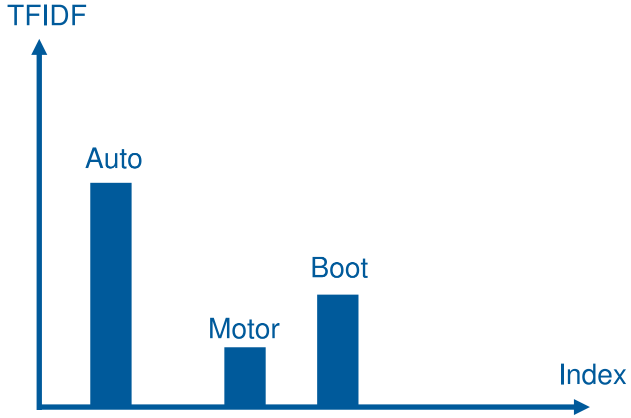
Document B	Document A
TFIDF word 1 TFIDF word 2 TFIDF word 3	TFIDF word 1 TFIDF word 2 TFIDF word 3

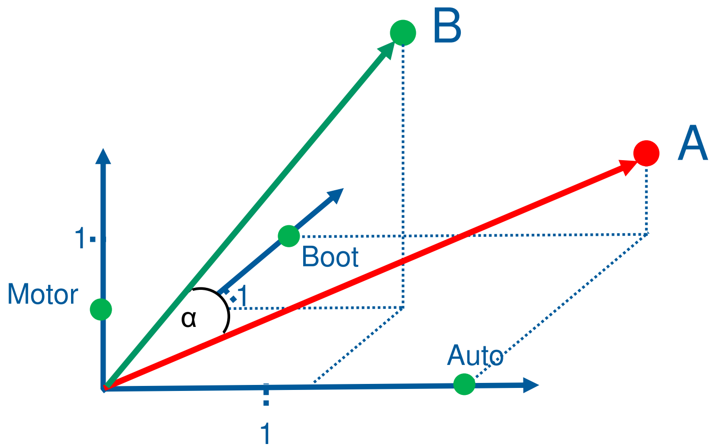
Word Embedding

Documents as Vectors

The similarity between documents is measured by the cosine similarity of the document vectors.

We are using TFIDF and BOW for Document Classification, successfully.





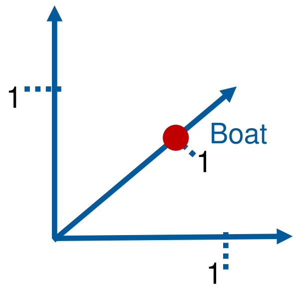
$$\begin{pmatrix} \text{TFIDF word 1} \\ \text{TFIDF word 2} \\ \text{TFIDF word 3} \end{pmatrix} \quad \begin{pmatrix} \text{TFIDF word 1} \\ \text{TFIDF word 2} \\ \text{TFIDF word 3} \end{pmatrix}$$

Document B Document A

Word embedding is any of a set of language modeling and feature learning techniques in natural language processing where words from the vocabulary are mapped to vectors of real numbers.

$$\begin{pmatrix}1\\0\\0\end{pmatrix}$$

Boat

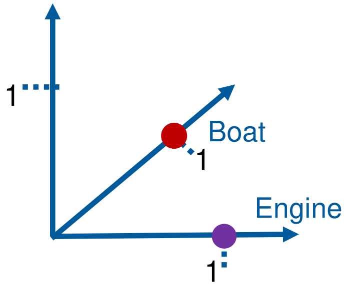


$$\begin{pmatrix}1\\0\\0\end{pmatrix}$$

Boat

Word embedding is any of a set of language modeling and feature learning techniques in natural language processing where words from the vocabulary are mapped to vectors of real numbers.

$$\underset{\text{Boat}}{\begin{bmatrix}1\\0\\0\end{bmatrix}} - \overset{\text{Engine}}{\begin{bmatrix}0\\1\\0\end{bmatrix}}$$



$$\underset{\text{Boat}}{\begin{bmatrix}1\\0\\0\end{bmatrix}} \bullet \overset{\text{Engine}}{\begin{bmatrix}0\\1\\0\end{bmatrix}}$$

Word Embedding

Words as Vectors

Word embedding is any of a set of language modeling and feature learning techniques in natural language processing where words from the vocabulary are mapped to vectors of real numbers.

Engine

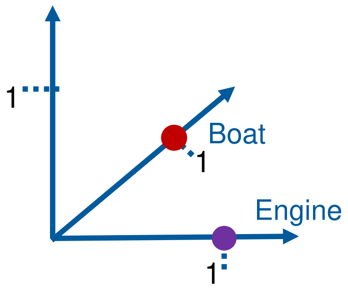
$$\begin{bmatrix}1\\0\\0\end{bmatrix} - \begin{bmatrix}0\\1\\0\end{bmatrix} = \begin{bmatrix}1\\-1\\0\end{bmatrix}$$

Boat

Engine

$$\begin{bmatrix}1\\0\\0\end{bmatrix} \bullet \begin{bmatrix}0\\1\\0\end{bmatrix} = 0$$

Boat



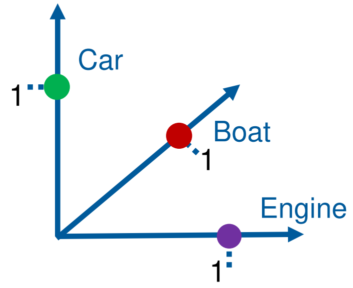
Word Embedding

Words as Vectors

Word embedding is any of a set of language modeling and feature learning techniques in natural language processing where words from the vocabulary are mapped to vectors of real numbers.

$$\underset{\text{Boat}}{\begin{pmatrix}1\\0\\0\end{pmatrix}} - \overset{\text{Engine}}{\begin{pmatrix}0\\1\\0\end{pmatrix}} = \begin{pmatrix}1\\-1\\0\end{pmatrix} \neq \underset{\text{Car}}{\begin{pmatrix}0\\0\\1\end{pmatrix}}$$

$$\underset{\text{Boat}}{\begin{pmatrix}1\\0\\0\end{pmatrix}} \bullet \overset{\text{Engine}}{\begin{pmatrix}0\\1\\0\end{pmatrix}} = \underset{\text{Car}}{\begin{pmatrix}0\\0\\1\end{pmatrix}}$$



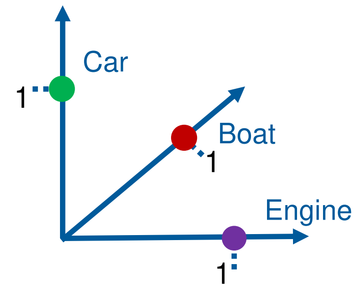
Word Embedding

Words as Vectors

Word embedding is any of a set of language modeling and feature learning techniques in natural language processing where words from the vocabulary are mapped to vectors of real numbers.

$$\underset{\text{Boat}}{\begin{pmatrix}1\\0\\0\end{pmatrix}} - \overset{\text{Engine}}{\begin{pmatrix}0\\1\\0\end{pmatrix}} = \begin{pmatrix}1\\-1\\0\end{pmatrix} \neq \underset{\text{Car}}{\begin{pmatrix}0\\0\\1\end{pmatrix}} - \overset{\text{Engine}}{\begin{pmatrix}0\\1\\0\end{pmatrix}} = \begin{pmatrix}0\\-1\\1\end{pmatrix}$$

$$\underset{\text{Boat}}{\begin{pmatrix}1\\0\\0\end{pmatrix}} \cdot \overset{\text{Engine}}{\begin{pmatrix}0\\1\\0\end{pmatrix}} = \underset{\text{Car}}{\begin{pmatrix}0\\0\\1\end{pmatrix}} \cdot \overset{\text{Engine}}{\begin{pmatrix}0\\1\\0\end{pmatrix}} = 0$$

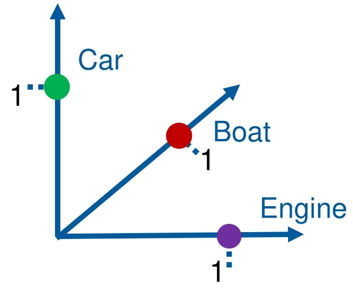


Word Embedding

Words as Vectors

Word embedding is any of a set of language modeling and feature learning techniques in natural language processing where words from the vocabulary are mapped to vectors of real numbers.

$$\underset{\text{Boat}}{\begin{bmatrix}1\\0\\0\end{bmatrix}} - \overset{\text{Engine}}{\begin{bmatrix}0\\1\\0\end{bmatrix}} = \begin{bmatrix}1\\-1\\0\end{bmatrix} \neq \underset{\text{Car}}{\begin{bmatrix}0\\0\\1\end{bmatrix}} - \overset{\text{Engine}}{\begin{bmatrix}0\\1\\0\end{bmatrix}} = \begin{bmatrix}0\\-1\\1\end{bmatrix}$$



$$\underset{\text{Boat}}{\begin{bmatrix}1\\0\\0\end{bmatrix}} \cdot \overset{\text{Engine}}{\begin{bmatrix}0\\1\\0\end{bmatrix}} = \underset{\text{Car}}{\begin{bmatrix}0\\0\\1\end{bmatrix}} \cdot \overset{\text{Engine}}{\begin{bmatrix}0\\1\\0\end{bmatrix}} = \overset{\text{Boat}}{\begin{bmatrix}1\\0\\0\end{bmatrix}} \cdot \underset{\text{Car}}{\begin{bmatrix}0\\0\\1\end{bmatrix}} = 0$$

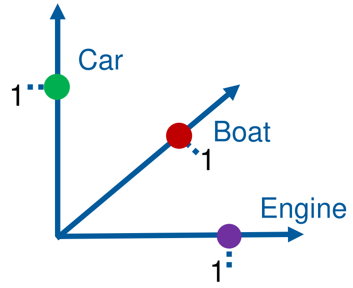
Word Embedding

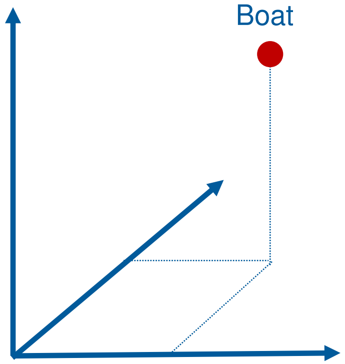
Words as Vectors

Word embedding is any of a set of language modeling and feature learning techniques in natural language processing where words from the vocabulary are mapped to vectors of real numbers.

$$\underset{\text{Boat}}{\begin{pmatrix}1\\0\\0\end{pmatrix}} - \overset{\text{Engine}}{\begin{pmatrix}0\\1\\0\end{pmatrix}} = \begin{pmatrix}1\\-1\\0\end{pmatrix} \neq \underset{\text{Car}}{\begin{pmatrix}0\\0\\1\end{pmatrix}} - \overset{\text{Engine}}{\begin{pmatrix}0\\1\\0\end{pmatrix}} = \begin{pmatrix}0\\-1\\1\end{pmatrix}$$

$$\underset{\text{Boat}}{\begin{pmatrix}1\\0\\0\end{pmatrix}} \bullet \overset{\text{Engine}}{\begin{pmatrix}0\\1\\0\end{pmatrix}} = \underset{\text{Car}}{\begin{pmatrix}0\\0\\1\end{pmatrix}} \bullet \overset{\text{Engine}}{\begin{pmatrix}0\\1\\0\end{pmatrix}} = \overset{\text{Boat}}{\begin{pmatrix}1\\0\\0\end{pmatrix}} \bullet \underset{\text{Car}}{\begin{pmatrix}0\\0\\1\end{pmatrix}} = 0$$





$$\underset{\text{Boat}}{\begin{pmatrix}0.11\\0.31\\0.46\end{pmatrix}}$$

$$\underset{\text{Boat}}{\begin{pmatrix}0.11\\0.31\\0.46\end{pmatrix}}$$

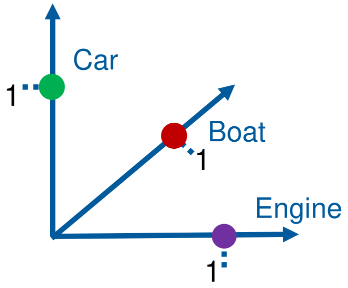
Word Embedding

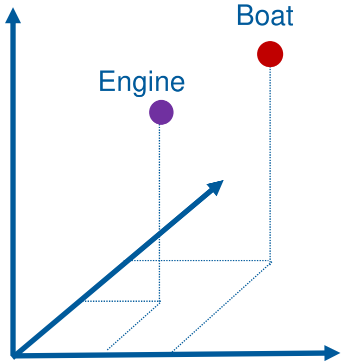
Words as Vectors

Word embedding is any of a set of language modeling and feature learning techniques in natural language processing where words from the vocabulary are mapped to vectors of real numbers.

$$\underset{\text{Boat}}{\begin{pmatrix}1\\0\\0\end{pmatrix}} - \overset{\text{Engine}}{\begin{pmatrix}0\\1\\0\end{pmatrix}} = \begin{pmatrix}1\\-1\\0\end{pmatrix} \neq \underset{\text{Car}}{\begin{pmatrix}0\\0\\1\end{pmatrix}} - \overset{\text{Engine}}{\begin{pmatrix}0\\1\\0\end{pmatrix}} = \begin{pmatrix}0\\-1\\1\end{pmatrix}$$

$$\underset{\text{Boat}}{\begin{pmatrix}1\\0\\0\end{pmatrix}} \bullet \overset{\text{Engine}}{\begin{pmatrix}0\\1\\0\end{pmatrix}} = \underset{\text{Car}}{\begin{pmatrix}0\\0\\1\end{pmatrix}} \bullet \overset{\text{Engine}}{\begin{pmatrix}0\\1\\0\end{pmatrix}} = \overset{\text{Boat}}{\begin{pmatrix}1\\0\\0\end{pmatrix}} \bullet \underset{\text{Car}}{\begin{pmatrix}0\\0\\1\end{pmatrix}} = 0$$





$$\underset{\text{Boat}}{\begin{pmatrix}0.11\\0.31\\0.46\end{pmatrix}} - \overset{\text{Engine}}{\begin{pmatrix}0.05\\0.01\\0.21\end{pmatrix}} = \begin{pmatrix}0.06\\0.30\\0.25\end{pmatrix}$$

$$\underset{\text{Boat}}{\begin{pmatrix}0.11\\0.31\\0.46\end{pmatrix}} \bullet \overset{\text{Engine}}{\begin{pmatrix}0.43\\0.01\\0.21\end{pmatrix}}$$

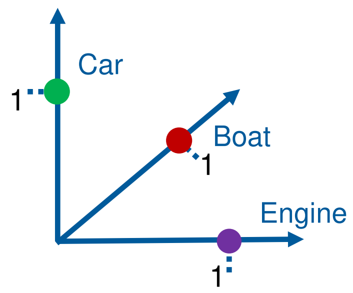
Word Embedding

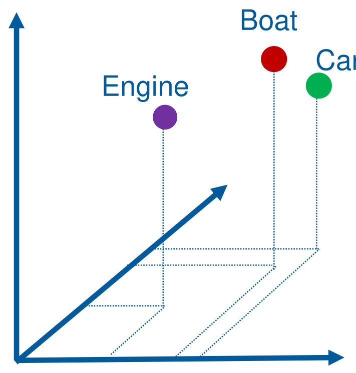
Words as Vectors

Word embedding is any of a set of language modeling and feature learning techniques in natural language processing where words from the vocabulary are mapped to vectors of real numbers.

$$\underset{\text{Boat}}{\begin{pmatrix}1\\0\\0\end{pmatrix}} - \overset{\text{Engine}}{\begin{pmatrix}0\\1\\0\end{pmatrix}} = \begin{pmatrix}1\\-1\\0\end{pmatrix} \neq \underset{\text{Car}}{\begin{pmatrix}0\\0\\1\end{pmatrix}} - \overset{\text{Engine}}{\begin{pmatrix}0\\1\\0\end{pmatrix}} = \begin{pmatrix}0\\-1\\1\end{pmatrix}$$

$$\underset{\text{Boat}}{\begin{pmatrix}1\\0\\0\end{pmatrix}} \bullet \overset{\text{Engine}}{\begin{pmatrix}0\\1\\0\end{pmatrix}} = \underset{\text{Car}}{\begin{pmatrix}0\\0\\1\end{pmatrix}} \bullet \overset{\text{Engine}}{\begin{pmatrix}0\\1\\0\end{pmatrix}} = \overset{\text{Boat}}{\begin{pmatrix}1\\0\\0\end{pmatrix}} \bullet \underset{\text{Car}}{\begin{pmatrix}0\\0\\1\end{pmatrix}} = 0$$





$$\underset{\text{Boat}}{\begin{pmatrix}0.11\\0.31\\0.46\end{pmatrix}} - \overset{\text{Engine}}{\begin{pmatrix}0.05\\0.01\\0.21\end{pmatrix}} = \begin{pmatrix}0.06\\0.30\\0.25\end{pmatrix} \quad \underset{\text{Car}}{\begin{pmatrix}0.22\\0.34\\0.45\end{pmatrix}}$$

$$\underset{\text{Boat}}{\begin{pmatrix}0.11\\0.31\\0.46\end{pmatrix}} \bullet \overset{\text{Engine}}{\begin{pmatrix}0.43\\0.01\\0.21\end{pmatrix}} \quad \underset{\text{Car}}{\begin{pmatrix}0.22\\0.34\\0.45\end{pmatrix}}$$

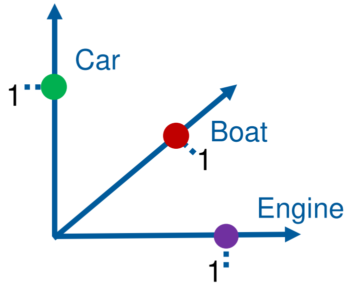
Word Embedding

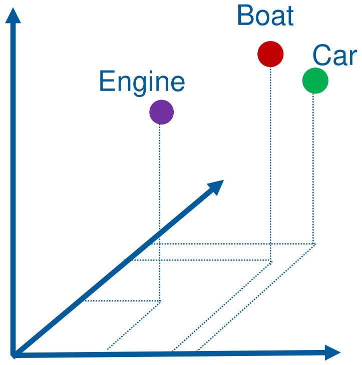
Words as Vectors

Word embedding is any of a set of language modeling and feature learning techniques in natural language processing where words from the vocabulary are mapped to vectors of real numbers.

$$\underset{\text{Boat}}{\begin{pmatrix}1\\0\\0\end{pmatrix}} - \overset{\text{Engine}}{\begin{pmatrix}0\\1\\0\end{pmatrix}} = \begin{pmatrix}1\\-1\\0\end{pmatrix} \neq \underset{\text{Car}}{\begin{pmatrix}0\\0\\1\end{pmatrix}} - \overset{\text{Engine}}{\begin{pmatrix}0\\1\\0\end{pmatrix}} = \begin{pmatrix}0\\-1\\1\end{pmatrix}$$

$$\underset{\text{Boat}}{\begin{pmatrix}1\\0\\0\end{pmatrix}} \cdot \overset{\text{Engine}}{\begin{pmatrix}0\\1\\0\end{pmatrix}} = \underset{\text{Car}}{\begin{pmatrix}0\\0\\1\end{pmatrix}} \cdot \overset{\text{Engine}}{\begin{pmatrix}0\\1\\0\end{pmatrix}} = \overset{\text{Boat}}{\begin{pmatrix}1\\0\\0\end{pmatrix}} \cdot \underset{\text{Car}}{\begin{pmatrix}0\\0\\1\end{pmatrix}} = 0$$





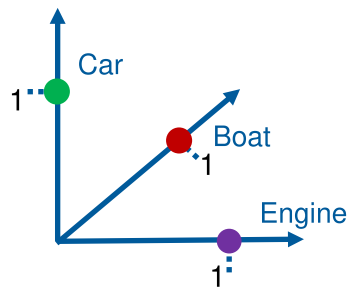
$$\underset{\text{Boat}}{\begin{pmatrix}0.11\\0.31\\0.46\end{pmatrix}} - \overset{\text{Engine}}{\begin{pmatrix}0.05\\0.01\\0.21\end{pmatrix}} = \begin{pmatrix}0.06\\0.30\\0.25\end{pmatrix} \neq \underset{\text{Car}}{\begin{pmatrix}0.22\\0.34\\0.45\end{pmatrix}} - \overset{\text{Engine}}{\begin{pmatrix}0.05\\0.01\\0.21\end{pmatrix}} = \begin{pmatrix}0.17\\0.33\\0.24\end{pmatrix}$$

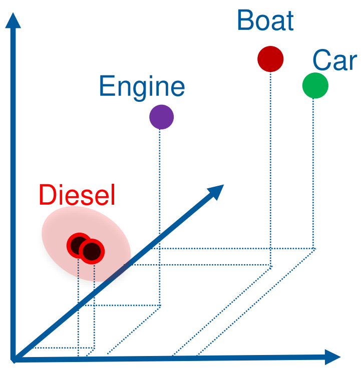
$$\underset{\text{Boat}}{\begin{pmatrix}0.11\\0.31\\0.46\end{pmatrix}} \cdot \overset{\text{Engine}}{\begin{pmatrix}0.43\\0.01\\0.21\end{pmatrix}} \neq \underset{\text{Car}}{\begin{pmatrix}0.22\\0.34\\0.45\end{pmatrix}} \cdot \overset{\text{Engine}}{\begin{pmatrix}0.43\\0.01\\0.21\end{pmatrix}} \neq \overset{\text{Boat}}{\begin{pmatrix}0.11\\0.31\\0.46\end{pmatrix}} \cdot \underset{\text{Car}}{\begin{pmatrix}0.22\\0.34\\0.45\end{pmatrix}} \neq 0$$

Word embedding is any of a set of language modeling and feature learning techniques in natural language processing where words from the vocabulary are mapped to vectors of real numbers.

$$\underset{\text{Boat}}{\begin{pmatrix}1\\0\\0\end{pmatrix}} - \overset{\text{Engine}}{\begin{pmatrix}0\\1\\0\end{pmatrix}} = \begin{bmatrix}1\\-1\\0\end{bmatrix} \neq \underset{\text{Car}}{\begin{pmatrix}0\\0\\1\end{pmatrix}} - \overset{\text{Engine}}{\begin{pmatrix}0\\1\\0\end{pmatrix}} = \begin{bmatrix}0\\-1\\1\end{bmatrix}$$

$$\underset{\text{Boat}}{\begin{pmatrix}1\\0\\0\end{pmatrix}} \cdot \overset{\text{Engine}}{\begin{pmatrix}0\\1\\0\end{pmatrix}} = \underset{\text{Car}}{\begin{pmatrix}0\\0\\1\end{pmatrix}} \cdot \overset{\text{Engine}}{\begin{pmatrix}0\\1\\0\end{pmatrix}} = \overset{\text{Boat}}{\begin{pmatrix}1\\0\\0\end{pmatrix}} \cdot \underset{\text{Car}}{\begin{pmatrix}0\\0\\1\end{pmatrix}} = 0$$





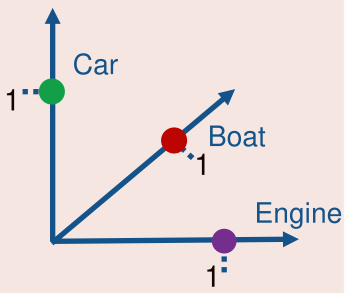
$$\underset{\text{Boat}}{\begin{pmatrix}0.11\\0.31\\0.46\end{pmatrix}} - \overset{\text{Engine}}{\begin{pmatrix}0.05\\0.01\\0.21\end{pmatrix}} = \overset{\text{Close to Diesel}}{\begin{pmatrix}0.06\\0.30\\0.25\end{pmatrix}} \neq \underset{\text{Car}}{\begin{pmatrix}0.22\\0.34\\0.45\end{pmatrix}} - \overset{\text{Engine}}{\begin{pmatrix}0.05\\0.01\\0.21\end{pmatrix}} = \overset{\text{Close to Diesel}}{\begin{pmatrix}0.17\\0.33\\0.24\end{pmatrix}}$$

$$\underset{\text{Boat}}{\begin{pmatrix}0.11\\0.31\\0.46\end{pmatrix}} \cdot \overset{\text{Engine}}{\begin{pmatrix}0.43\\0.01\\0.21\end{pmatrix}} \neq \underset{\text{Car}}{\begin{pmatrix}0.22\\0.34\\0.45\end{pmatrix}} \cdot \overset{\text{Engine}}{\begin{pmatrix}0.43\\0.01\\0.21\end{pmatrix}} \neq \overset{\text{Boat}}{\begin{pmatrix}0.11\\0.31\\0.46\end{pmatrix}} \cdot \underset{\text{Car}}{\begin{pmatrix}0.22\\0.34\\0.45\end{pmatrix}} \neq 0$$

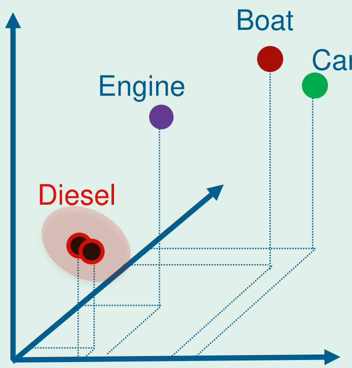
Word embedding is any of a set of language modeling and feature learning techniques in natural language processing where words from the vocabulary are mapped to vectors of real numbers.

$$\underset{\text{Boat}}{\begin{pmatrix}1\\0\\0\end{pmatrix}} - \overset{\text{Engine}}{\begin{pmatrix}0\\1\\0\end{pmatrix}} = \begin{bmatrix}1\\-1\\0\end{bmatrix} \neq \underset{\text{Car}}{\begin{pmatrix}0\\0\\1\end{pmatrix}} - \overset{\text{Engine}}{\begin{pmatrix}0\\1\\0\end{pmatrix}} = \begin{bmatrix}0\\-1\\1\end{bmatrix}$$

$$\underset{\text{Boat}}{\begin{pmatrix}1\\0\\0\end{pmatrix}} \cdot \overset{\text{Engine}}{\begin{pmatrix}0\\1\\0\end{pmatrix}} = \underset{\text{Car}}{\begin{pmatrix}0\\0\\1\end{pmatrix}} \cdot \overset{\text{Engine}}{\begin{pmatrix}0\\1\\0\end{pmatrix}} = \overset{\text{Boat}}{\begin{pmatrix}1\\0\\0\end{pmatrix}} \cdot \underset{\text{Car}}{\begin{pmatrix}0\\0\\1\end{pmatrix}} = 0$$



at most 3 words



maximal number of words?

$$\underset{\text{Boat}}{\begin{pmatrix}0.11\\0.31\\0.46\end{pmatrix}} - \overset{\text{Engine}}{\begin{pmatrix}0.05\\0.01\\0.21\end{pmatrix}} = \overset{\text{Close to Diesel}}{\begin{pmatrix}0.06\\0.30\\0.25\end{pmatrix}} \neq \underset{\text{Car}}{\begin{pmatrix}0.22\\0.34\\0.45\end{pmatrix}} - \overset{\text{Engine}}{\begin{pmatrix}0.05\\0.01\\0.21\end{pmatrix}} = \overset{\text{Close to Diesel}}{\begin{pmatrix}0.17\\0.33\\0.24\end{pmatrix}}$$

$$\underset{\text{Boat}}{\begin{pmatrix}0.11\\0.31\\0.46\end{pmatrix}} \cdot \overset{\text{Engine}}{\begin{pmatrix}0.43\\0.01\\0.21\end{pmatrix}} \neq \underset{\text{Car}}{\begin{pmatrix}0.22\\0.34\\0.45\end{pmatrix}} \cdot \overset{\text{Engine}}{\begin{pmatrix}0.43\\0.01\\0.21\end{pmatrix}} \neq \overset{\text{Boat}}{\begin{pmatrix}0.11\\0.31\\0.46\end{pmatrix}} \cdot \underset{\text{Car}}{\begin{pmatrix}0.22\\0.34\\0.45\end{pmatrix}} \neq 0$$

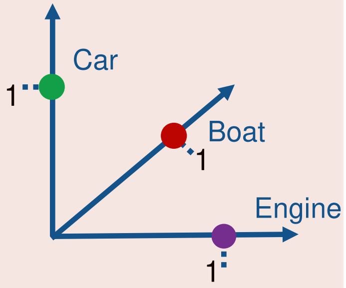
Word Embedding

Words as Vectors

Word embedding is any of a set of language modeling and feature learning techniques in natural language processing where words from the vocabulary are mapped to vectors of real numbers.

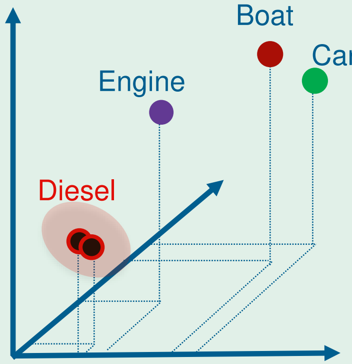
$$\underset{\text{Boat}}{\begin{pmatrix}1\\0\\0\end{pmatrix}} - \overset{\text{Engine}}{\begin{pmatrix}0\\1\\0\end{pmatrix}} = \begin{pmatrix}1\\-1\\0\end{pmatrix} \neq \underset{\text{Car}}{\begin{pmatrix}0\\0\\1\end{pmatrix}} - \overset{\text{Engine}}{\begin{pmatrix}0\\1\\0\end{pmatrix}} = \begin{pmatrix}0\\-1\\1\end{pmatrix}$$

$$\underset{\text{Boat}}{\begin{pmatrix}1\\0\\0\end{pmatrix}} \cdot \overset{\text{Engine}}{\begin{pmatrix}0\\1\\0\end{pmatrix}} = \underset{\text{Car}}{\begin{pmatrix}0\\0\\1\end{pmatrix}} \cdot \overset{\text{Engine}}{\begin{pmatrix}0\\1\\0\end{pmatrix}} = \overset{\text{Boat}}{\begin{pmatrix}1\\0\\0\end{pmatrix}} \cdot \underset{\text{Car}}{\begin{pmatrix}0\\0\\1\end{pmatrix}} = 0$$



at most 3 words

?

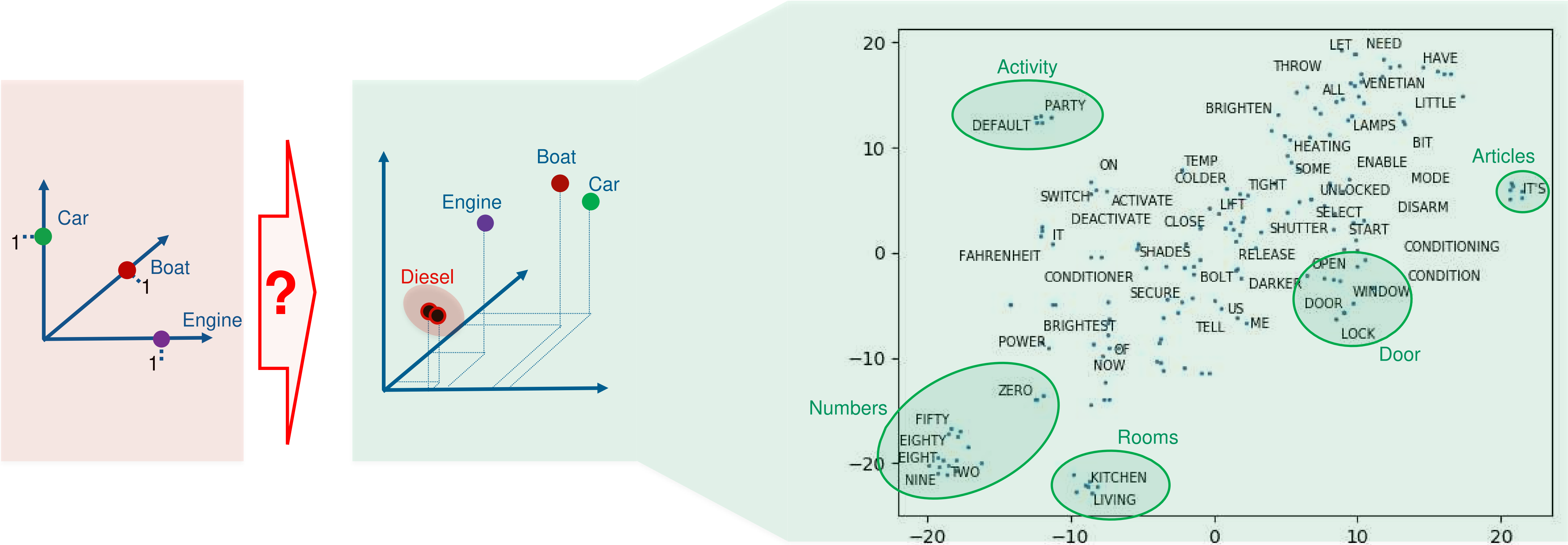


maximal number of words?

$$\underset{\text{Boat}}{\begin{pmatrix}0.11\\0.31\\0.46\end{pmatrix}} - \overset{\text{Engine}}{\begin{pmatrix}0.05\\0.01\\0.21\end{pmatrix}} = \overset{\text{Close to Diesel}}{\begin{pmatrix}0.06\\0.30\\0.25\end{pmatrix}} \neq \underset{\text{Car}}{\begin{pmatrix}0.22\\0.34\\0.45\end{pmatrix}} - \overset{\text{Engine}}{\begin{pmatrix}0.05\\0.01\\0.21\end{pmatrix}} = \overset{\text{Close to Diesel}}{\begin{pmatrix}0.17\\0.33\\0.24\end{pmatrix}}$$

$$\underset{\text{Boat}}{\begin{pmatrix}0.11\\0.31\\0.46\end{pmatrix}} \cdot \overset{\text{Engine}}{\begin{pmatrix}0.43\\0.01\\0.21\end{pmatrix}} \neq \underset{\text{Car}}{\begin{pmatrix}0.22\\0.34\\0.45\end{pmatrix}} \cdot \overset{\text{Engine}}{\begin{pmatrix}0.43\\0.01\\0.21\end{pmatrix}} \neq \overset{\text{Boat}}{\begin{pmatrix}0.11\\0.31\\0.46\end{pmatrix}} \cdot \underset{\text{Car}}{\begin{pmatrix}0.22\\0.34\\0.45\end{pmatrix}} \neq 0$$

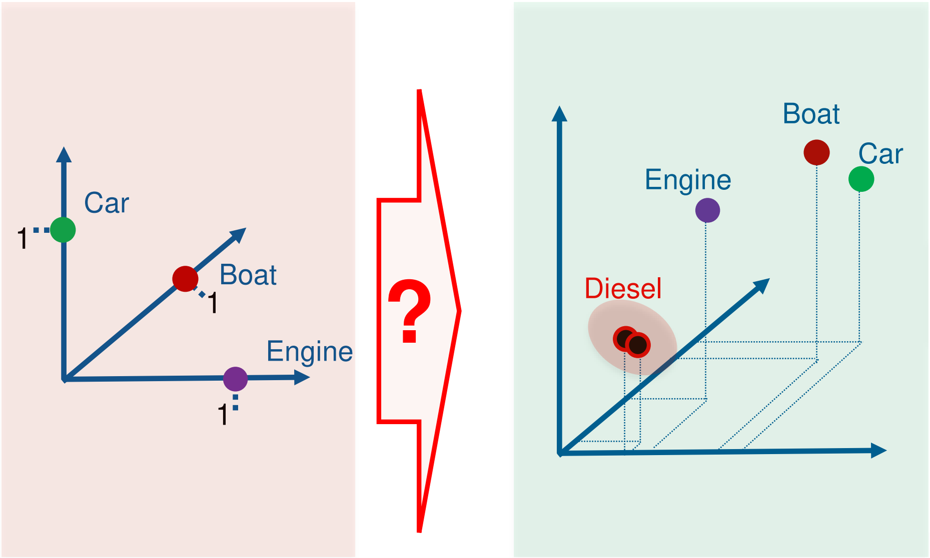
Word embedding is any of a set of language modeling and feature learning techniques in natural language processing where words from the vocabulary are mapped to vectors of real numbers.



Word Embedding

Transform a Word Vector

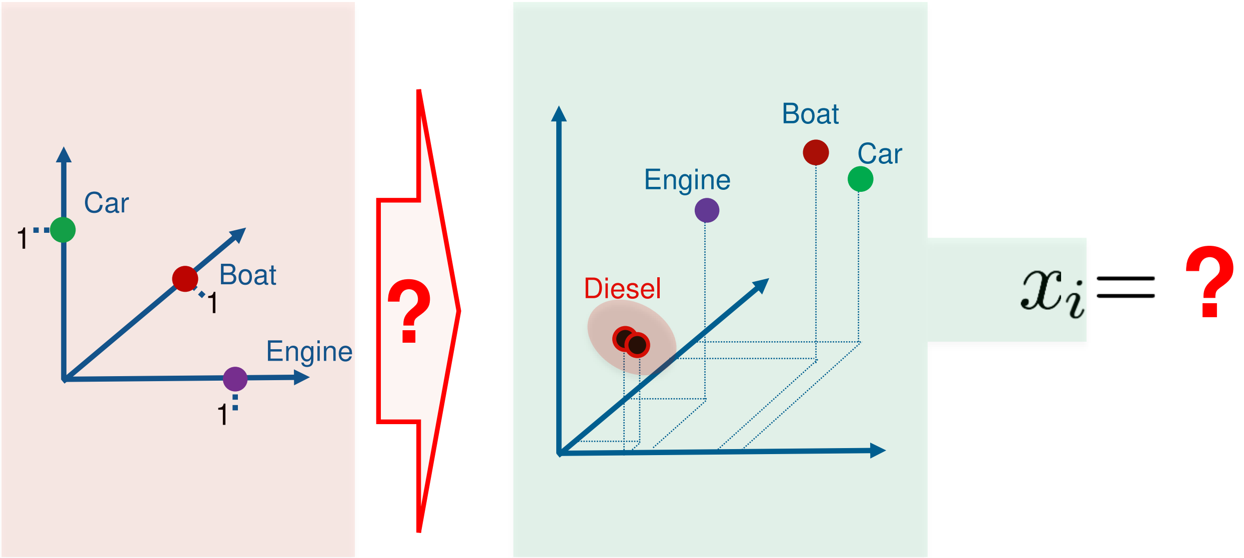
Word embedding is any of a set of language modeling and feature learning techniques in natural language processing where words from the vocabulary are mapped to vectors of real numbers.



Word Embedding

Transform a Word Vector

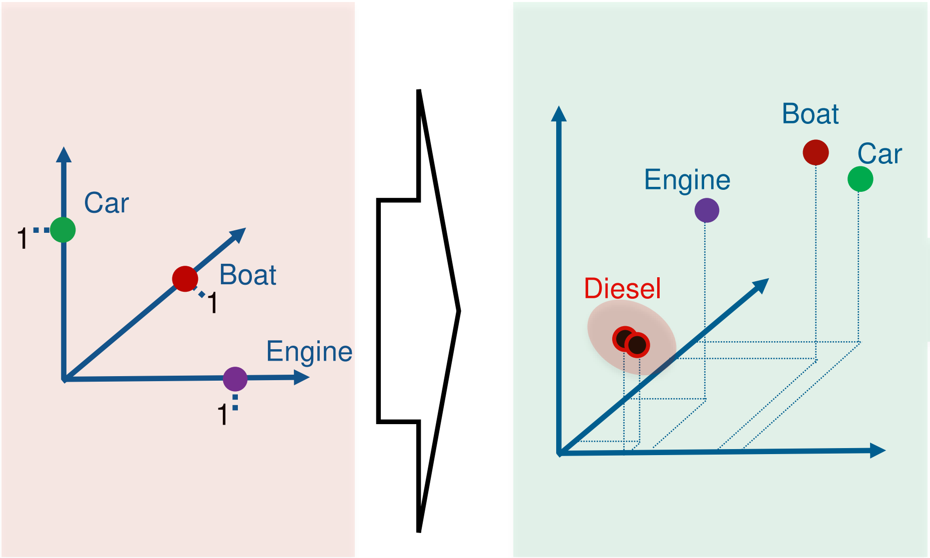
Word embedding is any of a set of language modeling and feature learning techniques in natural language processing where words from the vocabulary are mapped to vectors of real numbers.



Word Embedding

Transform a Word Vector

Word embedding is any of a set of language modeling and feature learning techniques in natural language processing where words from the vocabulary are mapped to vectors of real numbers.



$$x_i = E w_i$$

n dimensional
1-hot word vector

m x n dimensional
Embedding Matrix

Word Embedding

Transform a Word Vector

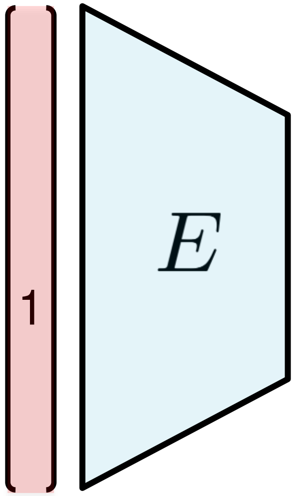
Word embedding is any of a set of language modeling and feature learning techniques in natural language processing where words from the vocabulary are mapped to vectors of real numbers.

w_i

$$\begin{bmatrix} \\ \\ \\ 1 \\ \\ \end{bmatrix}$$

Word embedding is any of a set of language modeling and feature learning techniques in natural language processing where words from the vocabulary are mapped to vectors of real numbers.

$$Ew_i$$

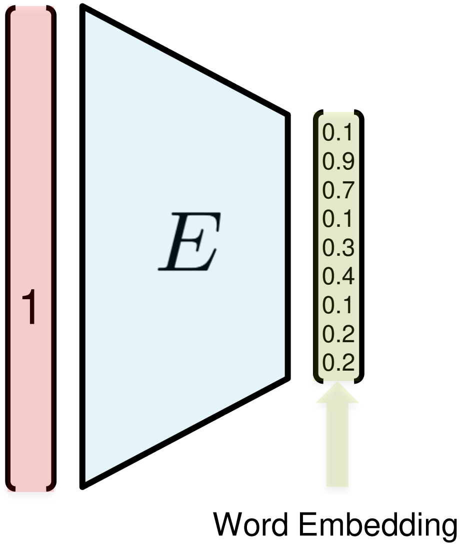


Word Embedding

Transform a Word Vector

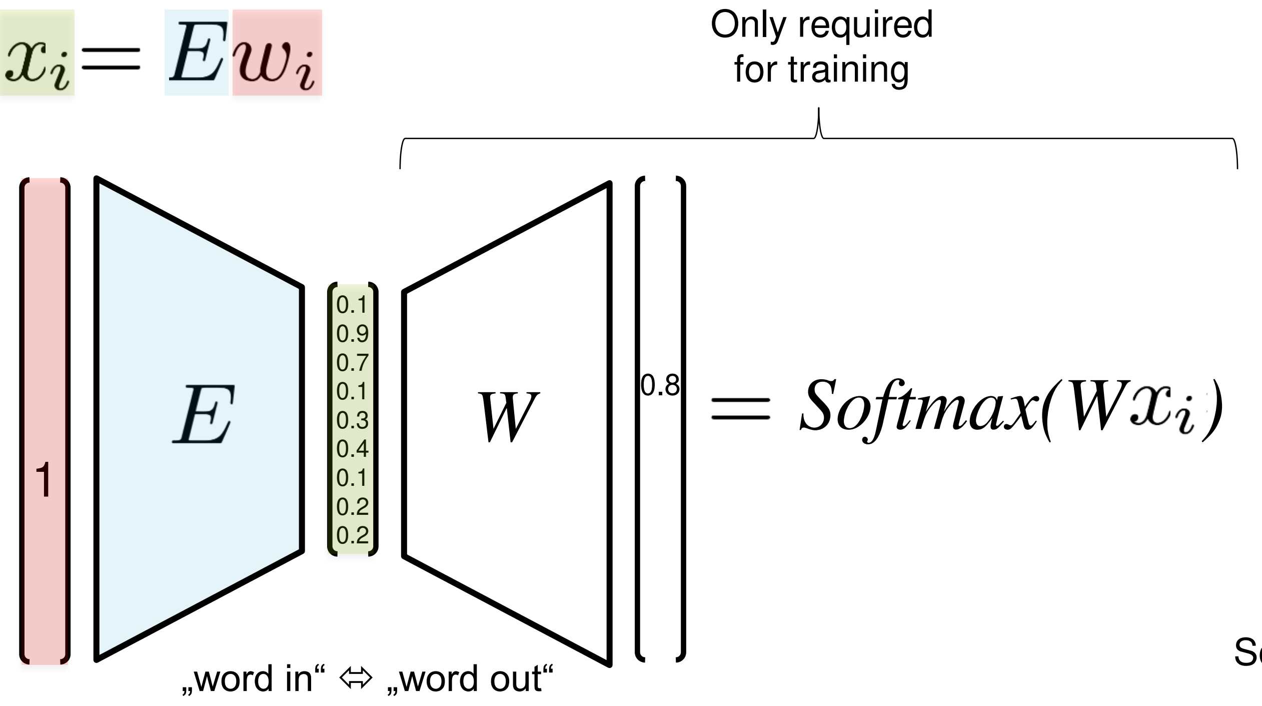
Word embedding is any of a set of language modeling and feature learning techniques in natural language processing where words from the vocabulary are mapped to vectors of real numbers.

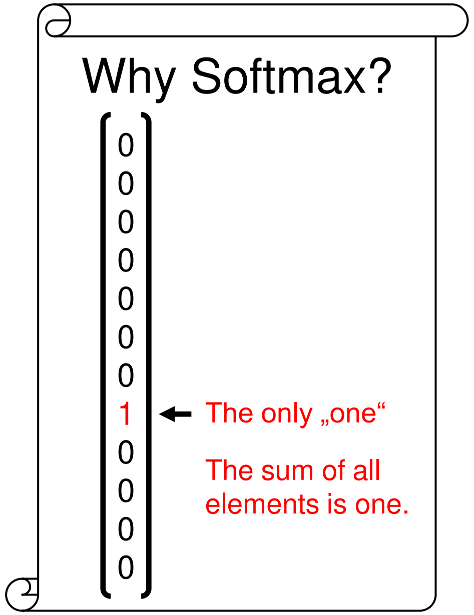
$$x_i = E w_i$$



Word embedding is any of a set of language modeling and feature learning techniques in natural language processing where words from the vocabulary are mapped to vectors of real numbers.

$$x_i = E w_i$$



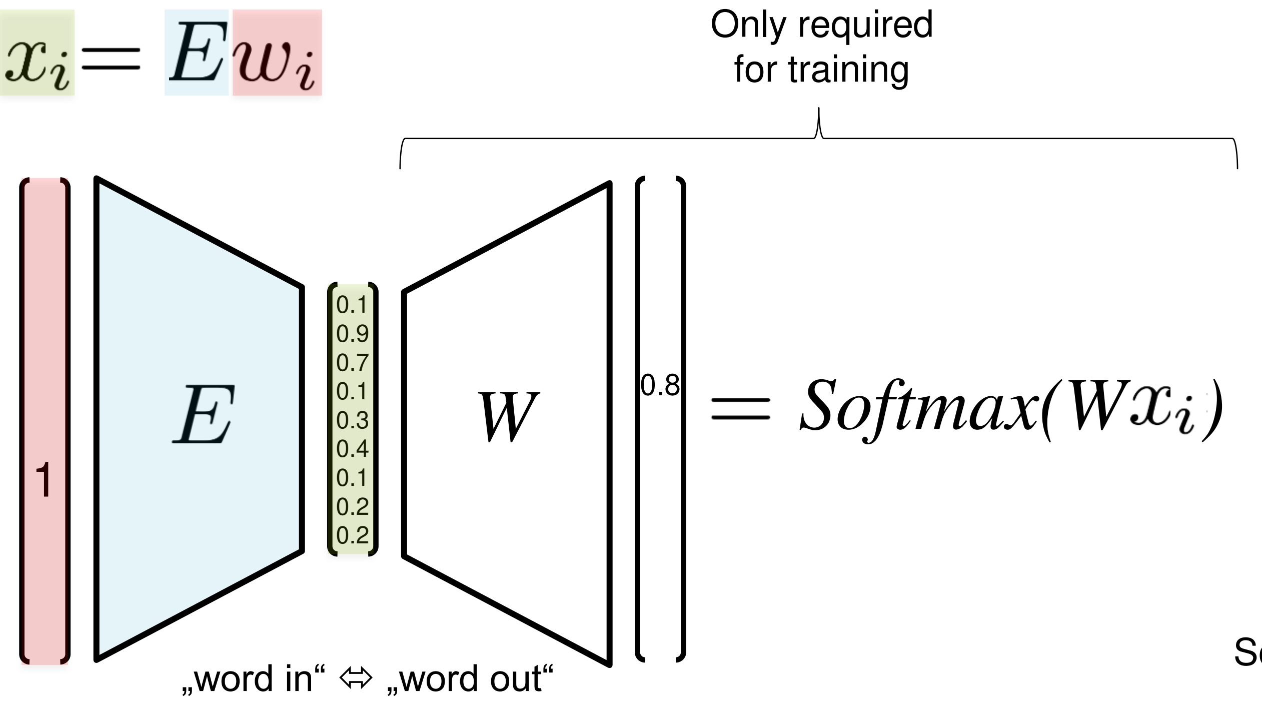


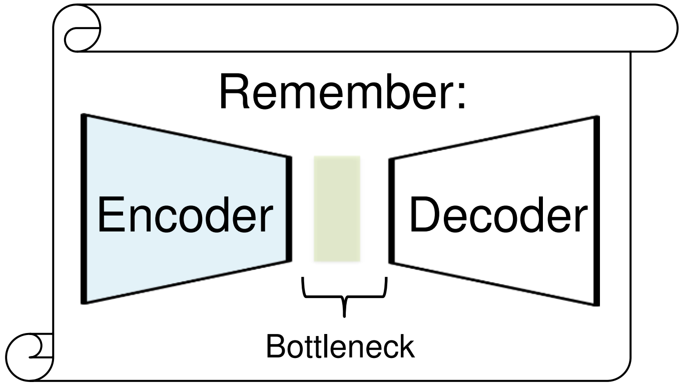
Softmax:

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \text{ for } i = 1, \dots, K \text{ and } \mathbf{z} = (z_1, \dots, z_K) \in \mathbb{R}^K$$

Word embedding is any of a set of language modeling and feature learning techniques in natural language processing where words from the vocabulary are mapped to vectors of real numbers.

$$x_i = E w_i$$



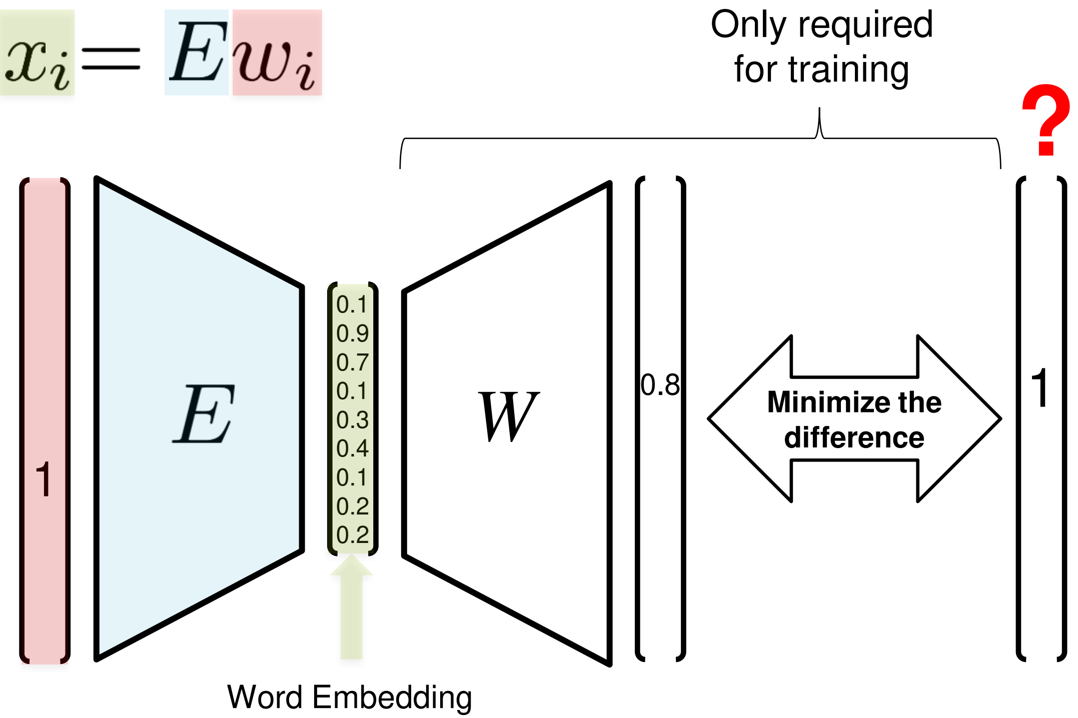


Softmax:

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \text{ for } i = 1, \dots, K \text{ and } \mathbf{z} = (z_1, \dots, z_K) \in \mathbb{R}^K$$

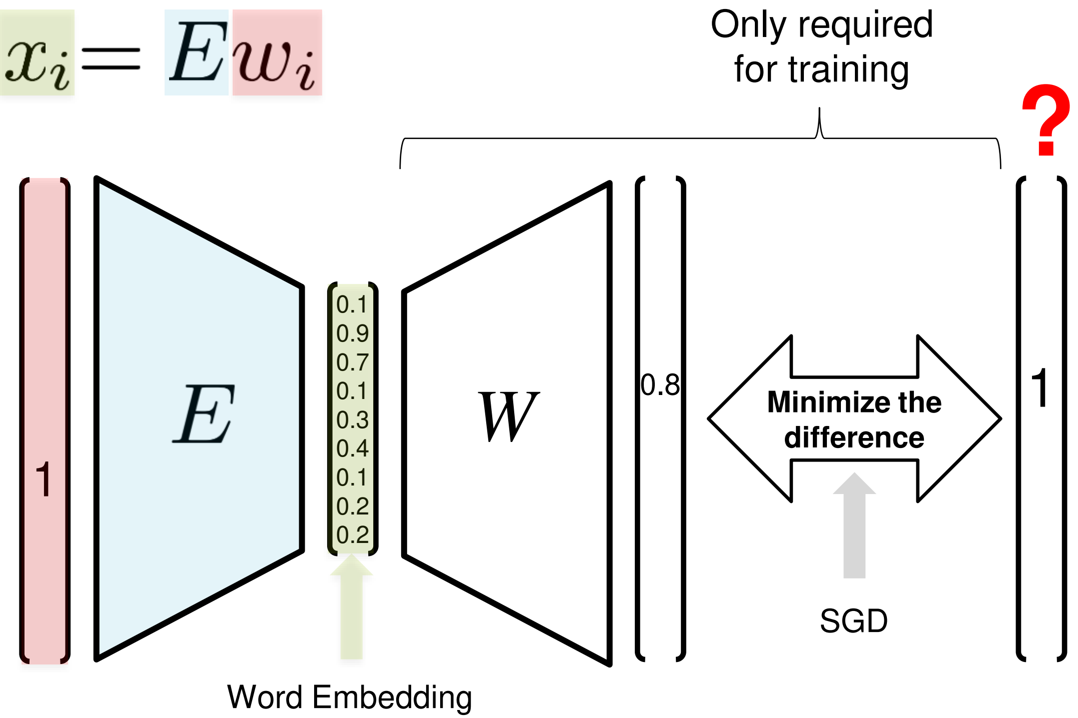
Word embedding is any of a set of language modeling and feature learning techniques in natural language processing where words from the vocabulary are mapped to vectors of real numbers.

$$x_i = E w_i$$



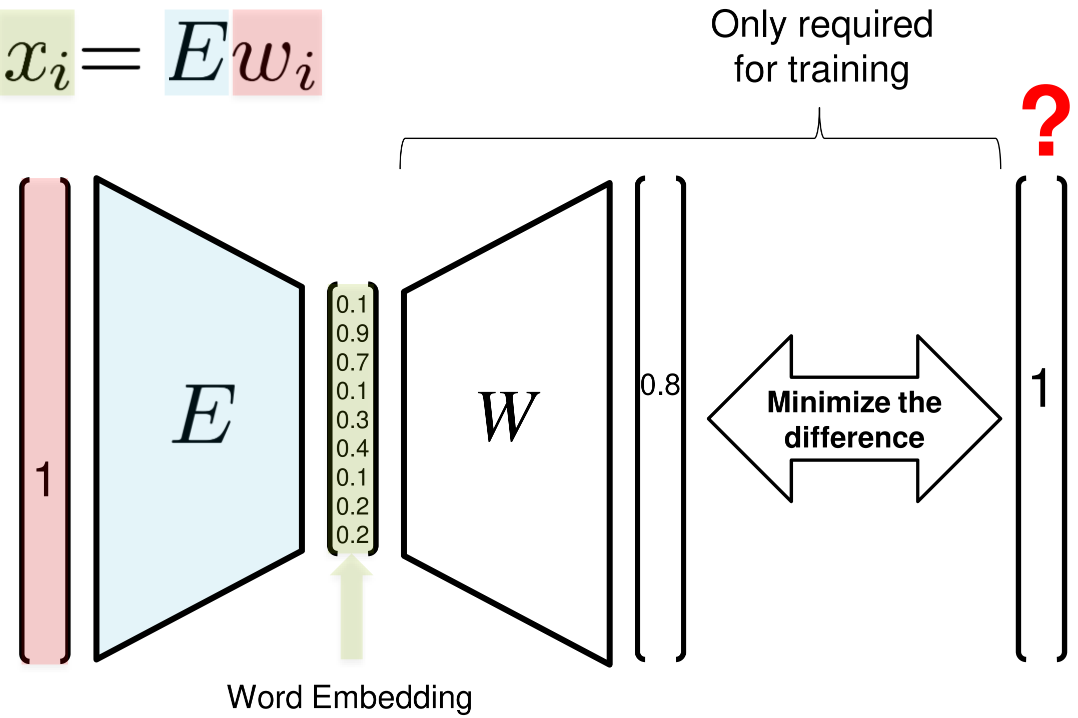
Word embedding is any of a set of language modeling and feature learning techniques in natural language processing where words from the vocabulary are mapped to vectors of real numbers.

$$x_i = E w_i$$



Word embedding is any of a set of language modeling and feature learning techniques in natural language processing where words from the vocabulary are mapped to vectors of real numbers.

$$x_i = E w_i$$



continuous Bag-Of-Words (cBOW)

skip-gram word embedding (word2vec)

Sparse vs. Dense vectors

Differences

tf-idf (or PMI) vectors are

- Long (length of vocabulary, 20.000 to 50.000)
- Sparse (most elements are zero)

Sparse vs. Dense vectors

Differences

tf-idf (or PMI) vectors are

- Long (length of vocabulary, 20.000 to 50.000)
- Sparse (most elements are zero)

Alternative: we want vectors that are

- Short (length 50-1000)
- Dense (most elements non-zero)

Sparse vs. Dense vectors

Question

tf-idf (or PMI) vectors are

- Long (length of vocabulary, 20.000 to 50.000)
- Sparse (most elements are zero)

Alternative: we want vectors that are

- Short (length 50-1000)
- Dense (most elements non-zero)

Why (short) dense vectors?

Sparse vs. Dense vectors

Motivation for dense vectors

tf-idf (or PMI) vectors are

- Long (length of vocabulary, 20.000 to 50.000)
- Sparse (most elements are zero)

Alternative: we want vectors that are

- Short (length 50-1000)
- Dense (most elements non-zero)

Why (short) dense vectors?

They may be **easier to use as features** in machine learning (fewer weights to tune)

- Dense vectors may **generalize better** than explicit counts
- Dense vectors may do **better at capturing synonymy**

They work better in practice

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

- Popular embedding method
- Very fast to train
- Code available on the web
- Idea: predict rather than count
- Word2vec provides various options.

SELF-SUPERVISION

- No need for human labels
- Bengio et al. (2003); Collobert et al. (2011)

Word2Vec

Skip Gram Idea: „Understanding ⇔ generating context“

Input: word
Output: its surrounding or *context words*

Skip Gram Idea: „Understanding ⇔ generating context“

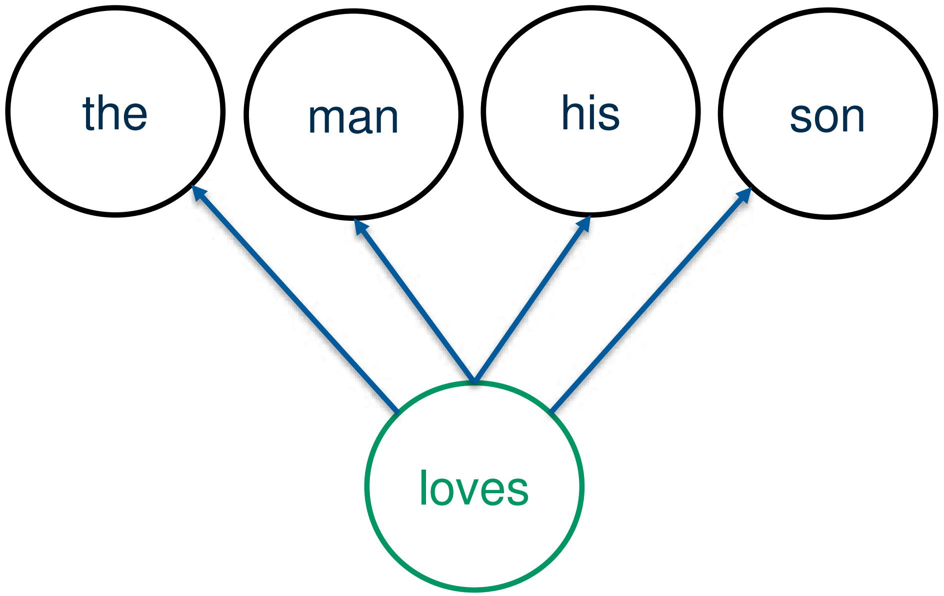
Input: word
Output: its surrounding or *context words*

Example: “the man loves his son”

Skip Gram Idea: „Understanding ⇔ generating context“

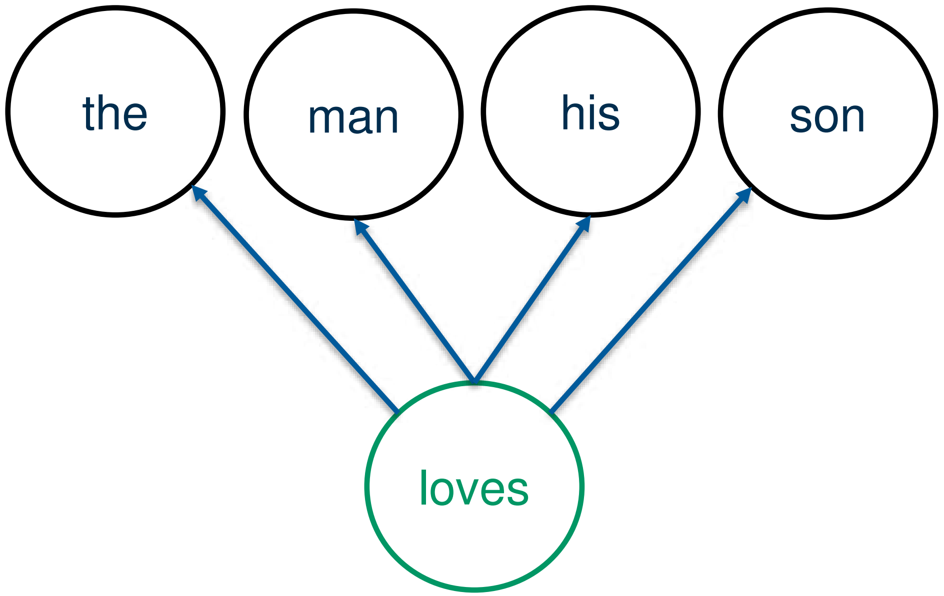
Input: word
Output: its surrounding or *context* words

Example: “the man loves his son”



Input: word
Output: its surrounding or *context* words

Example: “the man loves his son”



Skip gram considers conditional probabilities

$$P(\text{"the"}, \text{"man"}, \text{"his"}, \text{"son"} \mid \text{"loves"})$$

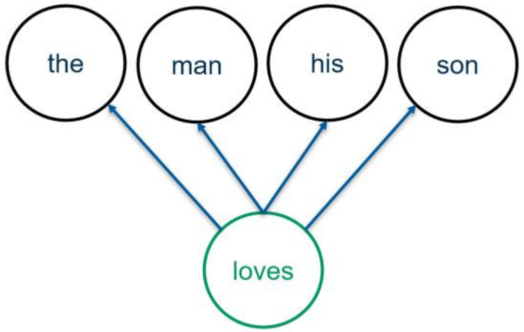
Word2vec: Skip Gram

Formalization

- $T = |X|$ length of text X
- $w^{(t)}$ = word at time step t
- m = context window size
- $\mathbf{v}_i \in \mathbb{R}^d$ = d-dimensional vector, when used as ***center word***
- $\boldsymbol{u}_i \in \mathbb{R}^d$ = d-dimensional vector, when used as ***context word***

Word2vec: Skip Gram

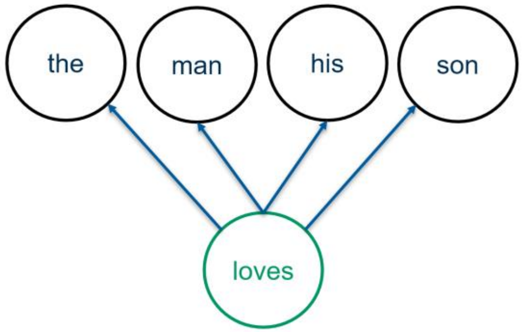
Formalization



- $T = |X|$ length of text X
- $w^{(t)}$ = word at time step t
- m = context window size
- $\mathbf{v}_i \in \mathbb{R}^d$ = d-dimensional vector, when used as ***center word***
- $\boldsymbol{u}_i \in \mathbb{R}^d$ = d-dimensional vector, when used as ***context word***

Word2vec: Skip Gram

Formalization



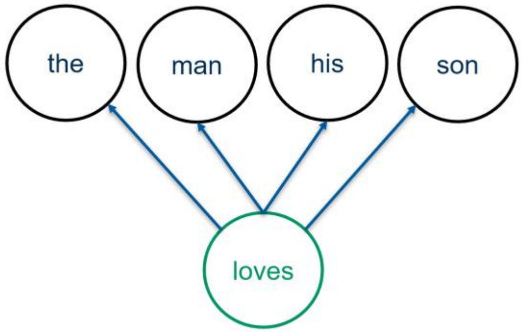
- $T = |X|$ length of text X
- $w^{(t)}$ = word at time step t
- m = context window size
- $\mathbf{v}_i \in \mathbb{R}^d$ = d-dimensional vector, when used as ***center word***
- $\boldsymbol{u}_i \in \mathbb{R}^d$ = d-dimensional vector, when used as ***context word***

$$P(\text{"the"}, \quad \text{"man"}, \quad \text{"his"}, \quad \text{"son"} \mid \text{"loves"})$$

$$\boldsymbol{w}^{(\boldsymbol{t}-\mathbf{2})} \quad \boldsymbol{w}^{(\boldsymbol{t}-\mathbf{1})} \quad \boldsymbol{w}^{(\boldsymbol{t}+\mathbf{1})} \quad \boldsymbol{w}^{(\boldsymbol{t}+\mathbf{2})} \quad \boldsymbol{w}^{(\boldsymbol{t})}$$

Word2vec: Skip Gram

Formalization



- $T = |X|$ length of text X
- $w^{(t)}$ = word at time step t
- m = context window size
- $\mathbf{v}_i \in \mathbb{R}^d$ = d-dimensional vector, when used as ***center word***
- $\boldsymbol{u}_i \in \mathbb{R}^d$ = d-dimensional vector, when used as ***context word***

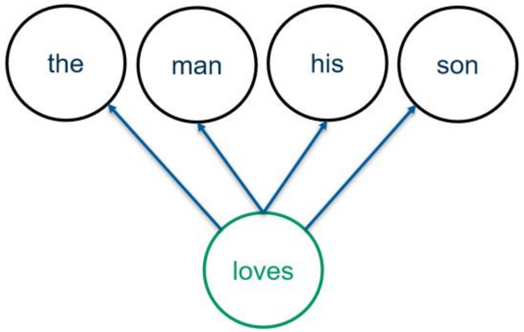
$$P(\text{"the"}, \text{"man"}, \text{"his"}, \text{"son"} \mid \text{"loves"}) \overset{*}{=} \mathrm{P}(\boldsymbol{w}^{(t-2)} \mid \boldsymbol{w}^{(t)}) \cdot \mathrm{P}(\boldsymbol{w}^{(t-1)} \mid \boldsymbol{w}^{(t)}) \cdot \mathrm{P}(\boldsymbol{w}^{(t+1)} \mid \boldsymbol{w}^{(t)}) \cdot \mathrm{P}(\boldsymbol{w}^{(t+2)} \mid \boldsymbol{w}^{(t)})$$

$$\boldsymbol{w}^{(t-2)} \quad \boldsymbol{w}^{(t-1)} \quad \boldsymbol{w}^{(t+1)} \quad \boldsymbol{w}^{(t+2)} \quad \boldsymbol{w}^{(t)}$$

** context words independently generated given any center word*

Word2vec: Skip Gram

Formalization



- $T = |X|$ length of text X
- $w^{(t)}$ = word at time step t
- m = context window size
- $\mathbf{v}_i \in \mathbb{R}^d$ = d-dimensional vector, when used as ***center word***
- $\boldsymbol{u}_i \in \mathbb{R}^d$ = d-dimensional vector, when used as ***context word***

$$P(\underset{\boldsymbol{w}^{(t-2)}}{\updownarrow}\text{"the"}, \underset{\boldsymbol{w}^{(t-1)}}{\updownarrow}\text{"man"}, \underset{\boldsymbol{w}^{(t+1)}}{\updownarrow}\text{"his"}, \underset{\boldsymbol{w}^{(t+2)}}{\updownarrow}\text{"son"} \mid \underset{\boldsymbol{w}^{(t)}}{\updownarrow}\text{"loves"}) \overset{*}{=} \mathrm{P}(\boldsymbol{w}^{(t-2)} \mid \boldsymbol{w}^{(t)}) \cdot \mathrm{P}(\boldsymbol{w}^{(t-1)} \mid \boldsymbol{w}^{(t)}) \cdot \mathrm{P}(\boldsymbol{w}^{(t+1)} \mid \boldsymbol{w}^{(t)}) \cdot \mathrm{P}(\boldsymbol{w}^{(t+2)} \mid \boldsymbol{w}^{(t)})$$

$$= \prod_{j=-2,\, j\neq 0}^{2} P(\boldsymbol{w}^{(t+j)} \mid \boldsymbol{w}^{(t)})$$

** context words independently generated given any center word*

Word2vec: Skip Gram

Formalization

- $T = |X|$ length of text X
- $w^{(t)}$ = word at time step t
- m = context window size
- $\mathbf{v}_i \in \mathbb{R}^d$ = d-dimensional vector, when used as ***center word***
- $\boldsymbol{u}_i \in \mathbb{R}^d$ = d-dimensional vector, when used as ***context word***

$$P(\text{"the"}, \ \text{"man"}, \ \text{"his"}, \ \text{"son"} \mid \text{"loves"}) \overset{*}{=} \mathrm{P}(\boldsymbol{w}^{(t-2)} \mid \boldsymbol{w}^{(t)}) \cdot \mathrm{P}(\boldsymbol{w}^{(t-1)} \mid \boldsymbol{w}^{(t)}) \cdot \mathrm{P}(\boldsymbol{w}^{(t+1)} \mid \boldsymbol{w}^{(t)}) \cdot \mathrm{P}(\boldsymbol{w}^{(t+2)} \mid \boldsymbol{w}^{(t)})$$

$$\boldsymbol{w}^{(t-2)} \quad \boldsymbol{w}^{(t-1)} \quad \boldsymbol{w}^{(t+1)} \quad \boldsymbol{w}^{(t+2)} \quad \boldsymbol{w}^{(t)}$$

$$= \prod_{j=-2,\ j\neq 0}^{2} P(\boldsymbol{w}^{(t+j)} \mid \boldsymbol{w}^{(t)})$$

** context words independently generated given any center word*

Word2vec: Skip Gram

Formalization

- $T = |X|$ length of text X
- $w^{(t)}$ = word at time step t
- m = context window size
- $\mathbf{v}_i \in \mathbb{R}^d$ = d-dimensional vector, when used as ***center word***
- $\boldsymbol{u}_i \in \mathbb{R}^d$ = d-dimensional vector, when used as ***context word***

Model parameters

$$\theta = \begin{bmatrix} v_{aardvark} \\ v_a \\ \vdots \\ v_{zebra} \\ u_{aardvark} \\ u_a \\ \vdots \\ u_{zebra} \end{bmatrix} \in \mathbb{R}^{2d|V|}$$

$$P(\text{"the"}, \text{"man"}, \text{"his"}, \text{"son"} \mid \text{"loves"}) \overset{*}{=} \mathrm{P}(\boldsymbol{w}^{(t-2)} \mid \boldsymbol{w}^{(t)}) \cdot \mathrm{P}(\boldsymbol{w}^{(t-1)} \mid \boldsymbol{w}^{(t)}) \cdot \mathrm{P}(\boldsymbol{w}^{(t+1)} \mid \boldsymbol{w}^{(t)}) \cdot \mathrm{P}(\boldsymbol{w}^{(t+2)} \mid \boldsymbol{w}^{(t)})$$

$$\boldsymbol{w}^{(t-2)} \quad \boldsymbol{w}^{(t-1)} \quad \boldsymbol{w}^{(t+1)} \quad \boldsymbol{w}^{(t+2)} \quad \boldsymbol{w}^{(t)}$$

$$= \prod_{j=-2,\, j\neq 0}^{2} P(\boldsymbol{w}^{(t+j)} \mid \boldsymbol{w}^{(t)})$$

** context words independently generated given any center word*

Skip Gram

Deriving the loss function

X = „This is an example text to learn word2vec algorithm. We can consider sentences like ***The man*** ***loves*** ***his son*** *so much that …*“

$\boldsymbol{w}^{(t)}$

$$\prod_{j=-2,\ j\neq 0}^{2} P\left(\boldsymbol{w}^{(t+j)} \middle| \boldsymbol{w}^{(t)}\right)$$

Skip Gram

Deriving the loss function

X = „This is an example text to learn word2vec algorithm. We can consider sentences like *The* ***man loves his son so*** *much that …*“

$\boldsymbol{w}^{(t+1)}$

$$\prod_{j=-2,\ j\neq 0}^{2} P\left(\boldsymbol{w}^{(t+j)} \middle| \boldsymbol{w}^{(t+1)}\right)$$

Skip Gram

Deriving the loss function

X = „This is an example text to learn word2vec algorithm. We can consider sentences like *The man* **loves his** **son** **so much** *that* …“

$\boldsymbol{w}^{(\boldsymbol{t+2})}$

$$\prod_{\boldsymbol{j}=-2,\, j\neq 0}^{2} P\left(\boldsymbol{w}^{(\boldsymbol{t}+\boldsymbol{j})} \middle| \boldsymbol{w}^{(\boldsymbol{t}+\mathbf{2})}\right)$$

Skip Gram

Deriving the loss function

X = „This is an example text to learn word2vec algorithm. We can consider sentences like ***The man*** ***loves*** ***his son*** *so much that …*“

$\boldsymbol{w}^{(t)}$

Idea: We want to have good word vectors for the whole text corpus X

Skip Gram

Deriving the loss function

X = „This is an example text to learn word2vec algorithm. We can consider sentences like ***The man loves his son*** *so much that …*“

$\boldsymbol{w}^{(t)}$

Idea: We want to have good word vectors for the whole text corpus X

$$maximize \quad \prod_{\text{center}} \prod_{\text{context}} \mathrm{P}(\text{ context} \mid \text{center })$$

Skip Gram

Deriving the loss function

X = „This is an example text to learn word2vec algorithm. We can consider sentences like ***The man loves his son*** *so much that …*“

$$\boldsymbol{w}^{(t)}$$

Idea: We want to have good word vectors for the whole text corpus X

$$maximize \qquad \prod_{\text{center}} \prod_{\text{context}} \mathrm{P}(\text{ context} \mid \text{center })$$

$$\Downarrow$$

$$maximize \qquad \prod_{t=1}^{T} \prod_{j=-m,\, j\neq 0}^{m} \mathrm{P}(\boldsymbol{w}^{(t+j)} | \boldsymbol{w}^{(t)})$$

X = „This is an example text to learn word2vec algorithm. We can consider sentences like ***The man loves his son*** *so much that …*“

$\boldsymbol{w}^{(t)}$

Idea: We want to have good word vectors for the whole text corpus X

$$\mathit{maximize} \quad \prod_{t=1}^{T} \prod_{j=-m,\ j\neq 0}^{m} \mathrm{P}(\boldsymbol{w}^{(t+j)} | \boldsymbol{w}^{(t)})$$

$$\Downarrow$$

$$\mathit{minimize} \quad -\prod_{t=1}^{T} \prod_{j=-m,\ j\neq 0}^{m} \mathrm{P}(\boldsymbol{w}^{(t+j)} | \boldsymbol{w}^{(t)})$$

Skip Gram

Deriving the loss function

X = „This is an example text to learn word2vec algorithm. We can consider sentences like ***The man loves his son*** *so much that* …“

$\boldsymbol{w}^{(t)}$

Idea: We want to have good word vectors for the whole text corpus X

$$minimize \quad -\prod_{t=1}^{T} \prod_{j=-m,\ j\neq 0}^{m} \mathrm{P}(\boldsymbol{w}^{(t+j)}|\boldsymbol{w}^{(t)})$$

log ⬇

$$minimize \quad -\sum_{t=1}^{T} \sum_{j=-m, j\neq 0}^{m} \log \mathrm{P}\left(\boldsymbol{w}^{(t+j)}\middle|\boldsymbol{w}^{(t)}\right)$$

NLLL:= **N**egative **L**og **L**ikelihood **L**oss

Skip Gram

Deriving the loss function

X = „This is an example text to learn word2vec algorithm. We can consider sentences like ***The man loves his son*** *so much that …*“

$\boldsymbol{w}^{(t)}$

Questions: - How to compute the probabilities?
- And how are probabilities and word vectors related?

$$minimize \quad -\sum_{t=1}^{T} \sum_{j=-m, j\neq 0}^{m} \log \mathrm{P}\left(\boldsymbol{w}^{(t+j)} \middle| \boldsymbol{w}^{(t)}\right)$$

Let $\mathrm{x} \in \mathbb{R}^n$. The *softmax* function is defined as

$$\mathrm{softmax}(\mathrm{x})_j = \frac{\exp(\mathrm{x_j})}{\sum_{i=0}^{n-1} \exp(x_i)}, \qquad j = 0, \dots, n-1$$

Question:

It is a function

$$\mathrm{softmax}\colon\ \mathbb{R}^{?} \to (a, b)^{?}$$

Let $\mathrm{x} \in \mathbb{R}^n$. The function $\mathrm{softmax} : \mathbb{R}^n \to (0,1)^n$ is defined as

$$\mathrm{softmax}(\mathrm{x})_j = \frac{\exp(\mathrm{x_j})}{\sum_{i=0}^{n-1} \exp(x_i)}, \qquad j = 0, \dots, n-1$$

Let $\mathrm{x} \in \mathbb{R}^n$. The function $\mathrm{softmax} : \mathbb{R}^n \rightarrow (0,1)^n$ is defined as

$$\mathrm{softmax}(\mathrm{x})_j = \frac{\exp(\mathrm{x_j})}{\sum_{i=0}^{n-1} \exp(x_i)}, \qquad j = 0, \dots, n-1$$

Question: How to compute the probabilities? And what‘s their relation to word vectors?

Let $\mathrm{x} \in \mathbb{R}^n$. The function $\mathrm{softmax} : \mathbb{R}^n \rightarrow (0,1)^n$ is defined as

$$\mathrm{softmax}(\mathrm{x})_j = \frac{\exp(\mathrm{x_j})}{\sum_{i=0}^{n-1} \exp(x_i)}, \qquad j = 0, \dots, n-1$$

Question: How to compute the probabilities? And what‘s their relation to word vectors?

Set $\mathrm{x}_j = \boldsymbol{u}_o^T \mathbf{v}_c$ and assume $n = |\mathrm{V}| \Leftrightarrow$ vocabulary size

=> $$\mathrm{P}\left(\boldsymbol{w}^{(t+j)} | \boldsymbol{w}^{(t)}\right) = \mathrm{softmax}(\boldsymbol{u}_o^T \mathbf{v}_c) = \frac{\exp(\boldsymbol{u}_o^T \mathbf{v}_c)}{\sum_{i=0}^{n-1} \exp(\boldsymbol{u}_i^T \mathbf{v}_c)}$$

Let $\mathrm{x} \in \mathbb{R}^n$. The function $\mathrm{softmax} : \mathbb{R}^n \to (0,1)^n$ is defined as

$$\mathrm{softmax}(\mathrm{x})_j = \frac{\exp(\mathrm{x_j})}{\sum_{i=0}^{n-1} \exp(x_i)}, \qquad j = 0, \dots, n-1$$

Question: How to compute the probabilities? And what‘s their relation to word vectors?

Set $\mathrm{x}_j = \boxed{\boldsymbol{u}_o^T \mathbf{v}_c}$ and assume $n = |\mathrm{V}| \Leftrightarrow$ vocabulary size

=> $$\mathrm{P}\left(\boldsymbol{w}^{(t+j)} \middle| \boldsymbol{w}^{(t)}\right) = \mathrm{softmax}(\boldsymbol{u}_o^T \mathbf{v}_c) = \frac{\exp(\boldsymbol{u}_o^T \mathbf{v}_c)}{\sum_{i=0}^{n-1} \exp(\boldsymbol{u}_i^T \mathbf{v}_c)}$$

Alternative notations for dot product: $\boldsymbol{u}_o^T \mathbf{v}_c = < \boldsymbol{u}_o, \mathbf{v}_c > = \boldsymbol{u}_o \cdot \mathbf{v}_c = \sum_{k=1}^{d} \boldsymbol{u}_{o_k} \mathbf{v}_{c_k}$

Let $x \in \mathbb{R}^n$. The function $\mathrm{softmax} : \mathbb{R}^n \to (0,1)^n$ is defined as

$$\mathrm{softmax}(\mathrm{x})_j = \frac{\exp(\mathrm{x_j})}{\sum_{i=0}^{n-1}\exp(x_i)}, \qquad j = 0, \dots, n-1$$

Question: How to compute the probabilities? And what‘s their relation to word vectors?

Set $\mathrm{x}_j = \boxed{\boldsymbol{u}_o^T \mathbf{v}_c}$ and assume $n = |\mathrm{V}| \Leftrightarrow$ vocabulary size

d-dimensional word vectors

$$\Rightarrow \quad \mathrm{P}\left(\boldsymbol{w}^{(t+j)} \middle| \boldsymbol{w}^{(t)}\right) = \mathrm{softmax}(\boldsymbol{u}_o^T \mathbf{v}_c) = \frac{\exp(\boldsymbol{u}_o^T \mathbf{v}_c)}{\sum_{i=0}^{n-1}\exp(\boldsymbol{u}_i^T \mathbf{v}_c)}$$

Alternative notations for dot product: $\boldsymbol{u}_o^T \mathbf{v}_c = \langle \boldsymbol{u}_o, \mathbf{v}_c \rangle = \boxed{\boldsymbol{u}_o} \cdot \boxed{\mathbf{v}_c} = \sum_{k=1}^{d} \boldsymbol{u}_{o_k} \mathbf{v}_{c_k}$

Let $\mathrm{x} \in \mathbb{R}^n$. The function $\mathrm{softmax} : \mathbb{R}^n \to (0,1)^n$ is defined as

$$\mathrm{softmax}(\mathrm{x})_j = \frac{\exp(\mathrm{x_j})}{\sum_{i=0}^{n-1} \exp(x_i)}, \qquad j = 0, \dots, n-1$$

Question: How to compute the probabilities? And what‘s their relation to word vectors?

Set $\mathrm{x}_j = \boldsymbol{u}_o^T \mathbf{v}_c$ and assume $n = |\mathrm{V}| \Leftrightarrow$ vocabulary size

=> $$\mathrm{P}\left(\boldsymbol{w}^{\boxed{(t+j)}} \middle| \boldsymbol{w}^{\boxed{(t)}}\right) = \mathrm{softmax}(\boldsymbol{u}_{\boxed{o}}^T \mathbf{v}_{\boxed{c}}) = \frac{\exp(\boldsymbol{u}_o^T \mathbf{v}_c)}{\sum_{i=0}^{n-1} \exp(\boldsymbol{u}_i^T \mathbf{v}_c)}$$

word positions in running text

indices in vocabulary V

Skip Gram

Deriving the loss function

X = „This is an example text to learn word2vec algorithm. We can consider sentences like ***The man loves his son*** *so much that …*“

$\boldsymbol{w}^{(t)}$

Question: How to compute the probabilities?

$$\textit{minimize} \quad -\sum_{t=1}^{T} \sum_{j=-m, j\neq 0}^{m} \log \mathrm{P}\left(\boldsymbol{w}^{(t+j)} \middle| \boldsymbol{w}^{(t)}\right)$$

$$P(w_o \mid w_c) = \frac{\exp(\mathbf{u}_o^\top \mathbf{v}_c)}{\sum_{i \in \mathcal{V}} \exp(\mathbf{u}_i^\top \mathbf{v}_c)},$$

$$\mathbf{v}_c \coloneqq \textit{center word with index } c \in V$$

$$\boldsymbol{u}_i, \boldsymbol{u}_o \coloneqq \textit{context words with indices } i, o \in V$$

Vocabulary $V = \{0, 1, \dots, |V| - 1\}$

Skip Gram: Gradients

Minimizing the loss function

$$minimize \qquad -\sum_{t=1}^{T} \sum_{j=-m, j\neq 0}^{m} \log \mathrm{P}\left(\boldsymbol{w}^{(\boldsymbol{t}+\boldsymbol{j})} \middle| \boldsymbol{w}^{(\boldsymbol{t})}\right)$$

Vocabulary $V = \{0, 1, \dots, |V| - 1\}$

$\mathbf{v}_c :=$ *center word with index* $c \in V$
$\boldsymbol{u}_i :=$ *context word with index* $i \in V$

Skip Gram: Gradients

Minimizing the loss function

Theta holds all variables (center and context word vectors)

$$minimize \quad \underbrace{-\sum_{t=1}^{T} \sum_{j=-m, j\neq 0}^{m} \log \mathrm{P}\left(\boldsymbol{w}^{(t+j)} \middle| \boldsymbol{w}^{(t)}\right)}_{L(\theta)}$$

Vocabulary $V = \{0, 1, \ldots, |V| - 1\}$

$\mathbf{v}_c \coloneqq center\ word\ with\ index\ c \in V$

$\boldsymbol{u}_i \coloneqq context\ word\ with\ index\ i \in V$

Skip Gram: Gradients

Minimizing the loss function

Theta holds all variables (center and context word vectors)

$$\text{minimize} \quad \overbrace{-\sum_{t=1}^{T} \sum_{j=-m, j\neq 0}^{m} \log \mathrm{P}\left(\boldsymbol{w}^{(t+j)} \middle| \boldsymbol{w}^{(t)}\right)}^{L(\theta)}$$

Vocabulary $V = \{0, 1, \dots, |V| - 1\}$

$\mathbf{v}_c :=$ *center word with index* $c \in V$

$\boldsymbol{u}_i :=$ *context word with index* $i \in V$

Skip Gram: Gradients

Minimizing the loss function

Theta holds all variables (center and context word vectors)

$$\text{minimize} \quad \underbrace{-\sum_{t=1}^{T} \sum_{j=-m, j \neq 0}^{m} \log P\left(\boldsymbol{w}^{(t+j)} \middle| \boldsymbol{w}^{(t)}\right)}_{L(\theta)}$$

$$P(w_o \mid w_c) = \frac{\exp(\mathbf{u}_o^\top \mathbf{v}_c)}{\sum_{i \in \mathcal{V}} \exp(\mathbf{u}_i^\top \mathbf{v}_c)},$$

Question: What is the log of P(…) ?

Vocabulary $V = \{0, 1, \dots, |V| - 1\}$

$\mathbf{v}_c \coloneqq \text{center word with index } c \in V$

$\boldsymbol{u}_i \coloneqq \text{context word with index } i \in V$

Skip Gram: Gradients

Minimizing the loss function

Theta holds all variables (word vectors)

$$\textit{minimize} \quad \underbrace{- \sum_{t=1}^{T} \sum_{j=-m, j\neq 0}^{m} \log P\left(\boldsymbol{w}^{(t+j)} \middle| \boldsymbol{w}^{(t)}\right)}_{L(\theta)}$$

$$P(w_o \mid w_c) = \frac{\exp(\mathbf{u}_o^\top \mathbf{v}_c)}{\sum_{i\in\mathcal{V}} \exp(\mathbf{u}_i^\top \mathbf{v}_c)},$$

Question: What is the log of P(…) ?

$$\log P(w_o \mid w_c) = \mathbf{u}_o^\top \mathbf{v}_c - \log\left(\sum_{i\in\mathcal{V}} \exp(\mathbf{u}_i^\top \mathbf{v}_c)\right)$$

Vocabulary $V = \{0, 1, \dots, |V| - 1\}$

$\mathbf{v}_c \coloneqq$ *center word with index* $c \in V$
$\boldsymbol{u}_i \coloneqq$ *context word with index* $i \in V$

Skip Gram: Gradients

Minimizing the loss function

Theta holds all variables (word vectors)

$$minimize \quad \underbrace{-\sum_{t=1}^{T}\sum_{j=-m,j\neq 0}^{m} \log \mathrm{P}\left(\boldsymbol{w}^{(t+j)} \middle| \boldsymbol{w}^{(t)}\right)}_{L(\theta)}$$

$$P(w_o \mid w_c) = \frac{\exp(\mathbf{u}_o^\top \mathbf{v}_c)}{\sum_{i\in\mathcal{V}} \exp(\mathbf{u}_i^\top \mathbf{v}_c)},$$

Question: Which derivatives do we need to compute?

Vocabulary $V = \{0, 1, \ldots, |V| - 1\}$

$\mathbf{v}_c :=$ *center word with index* $c \in V$
$\boldsymbol{u}_i :=$ *context word with index* $i \in V$

Skip Gram: Gradients

Minimizing the loss function

Theta holds all variables (word vectors)

$$\text{minimize} \qquad \overbrace{-\sum_{t=1}^{T} \sum_{j=-m, j\neq 0}^{m} \log \mathrm{P}\left(\boldsymbol{w}^{(t+\boldsymbol{j})} \middle| \boldsymbol{w}^{(t)}\right)}^{L(\theta)}$$

$$P(w_o \mid w_c) = \frac{\exp(\mathbf{u}_o^\top \mathbf{v}_c)}{\sum_{i\in\mathcal{V}} \exp(\mathbf{u}_i^\top \mathbf{v}_c)},$$

Compute $\nabla_\theta L(\theta)$, where $\theta = \left(u_0, \dots, u_{|V|-1}, \mathbf{v}_0, \dots, \mathbf{v}_{|V|-1}\right)$, $u_i, \mathbf{v}_j \in \mathbb{R}^d$

Vocabulary $V = \{0, 1, \dots, |V| - 1\}$

$\mathbf{v}_c :=$ *center word with index* $c \in V$
$\boldsymbol{u}_i :=$ *context word with index* $i \in V$

Skip Gram: Gradients

Minimizing the loss function

Theta holds all variables (word vectors)

$$\textit{minimize} \qquad \overbrace{-\sum_{t=1}^{T}\sum_{j=-m, j\neq 0}^{m} \log \mathrm{P}\left(\boldsymbol{w}^{(t+j)} \middle| \boldsymbol{w}^{(t)}\right)}^{L(\theta)}$$

$$P(w_o \mid w_c) = \frac{\exp(\mathbf{u}_o^\top \mathbf{v}_c)}{\sum_{i\in\mathcal{V}} \exp(\mathbf{u}_i^\top \mathbf{v}_c)},$$

Compute $\nabla_\theta L(\theta)$, where $\theta = \left(u_0, \dots, u_{|V|-1}, \mathbf{v}_0, \dots, \mathbf{v}_{|V|-1}\right)$, $u_i, \mathbf{v}_j \in \mathbb{R}^d$

Derivative w.r.t. center word vector parameters:

$$\begin{aligned}
\frac{\partial \log P(w_o \mid w_c)}{\partial \mathbf{v}_c} &= \mathbf{u}_o - \frac{\sum_{j\in\mathcal{V}} \exp(\mathbf{u}_j^\top \mathbf{v}_c)\mathbf{u}_j}{\sum_{i\in\mathcal{V}} \exp(\mathbf{u}_i^\top \mathbf{v}_c)} \\
&= \mathbf{u}_o - \sum_{j\in\mathcal{V}} \left(\frac{\exp(\mathbf{u}_j^\top \mathbf{v}_c)}{\sum_{i\in\mathcal{V}} \exp(\mathbf{u}_i^\top \mathbf{v}_c)}\right)\mathbf{u}_j \\
&= \mathbf{u}_o - \sum_{j\in\mathcal{V}} P(w_j \mid w_c)\mathbf{u}_j.
\end{aligned}$$

Vocabulary $V = \{0, 1, \dots, |V| - 1\}$

$\mathbf{v}_c \coloneqq \textit{center word with index } c \in V$

$\boldsymbol{u}_i \coloneqq \textit{context word with index } i \in V$

Skip Gram: Gradients

Minimizing the loss function

Theta holds all variables (word vectors)

$$\text{minimize} \quad \overbrace{-\sum_{t=1}^{T}\sum_{j=-m,j\neq 0}^{m} \log \mathrm{P}\left(\boldsymbol{w}^{(t+j)} \middle| \boldsymbol{w}^{(t)}\right)}^{L(\theta)}$$

$$P(w_o \mid w_c) = \frac{\exp(\mathbf{u}_o^\top \mathbf{v}_c)}{\sum_{i\in\mathcal{V}} \exp(\mathbf{u}_i^\top \mathbf{v}_c)},$$

Compute $\nabla_\theta L(\theta)$, where $\theta = \left(u_0, \dots, u_{|V|-1}, \mathbf{v}_0, \dots, \mathbf{v}_{|V|-1}\right)$, $u_i, \mathbf{v}_j \in \mathbb{R}^d$

Derivative w.r.t. center word vector parameters:

$$\begin{aligned}
\frac{\partial \log P(w_o \mid w_c)}{\partial \mathbf{v}_c} &= \mathbf{u}_o - \frac{\sum_{j\in\mathcal{V}} \exp(\mathbf{u}_j^\top \mathbf{v}_c)\mathbf{u}_j}{\sum_{i\in\mathcal{V}} \exp(\mathbf{u}_i^\top \mathbf{v}_c)} \\
&= \mathbf{u}_o - \sum_{j\in\mathcal{V}} \left(\frac{\exp(\mathbf{u}_j^\top \mathbf{v}_c)}{\sum_{i\in\mathcal{V}} \exp(\mathbf{u}_i^\top \mathbf{v}_c)}\right)\mathbf{u}_j \\
&= \mathbf{u}_o - \underbrace{\sum_{j\in\mathcal{V}} P(w_j \mid w_c)\mathbf{u}_j}_{\text{Average over all context vectors, weighted by their respective probability}}.
\end{aligned}$$

Average over all context vectors, weighted by their respective probability

Vocabulary $V = \{0, 1, \dots, |V| - 1\}$

$\mathbf{v}_c \coloneqq$ *center word with index* $c \in V$
$\boldsymbol{u}_i \coloneqq$ *context word with index* $i \in V$

Skip Gram: Gradients

Minimizing the loss function

Theta holds all variables (word vectors)

$$\text{minimize} \quad \overbrace{-\sum_{t=1}^{T}\sum_{j=-m, j\neq 0}^{m} \log P\left(\boldsymbol{w}^{(t+j)} \middle| \boldsymbol{w}^{(t)}\right)}^{L(\theta)}$$

$$P(w_o \mid w_c) = \frac{\exp(\mathbf{u}_o^\top \mathbf{v}_c)}{\sum_{i\in\mathcal{V}} \exp(\mathbf{u}_i^\top \mathbf{v}_c)},$$

Compute $\nabla_\theta L(\theta)$, where $\theta = (u_0, \dots, u_{|V|-1}, \mathbf{v}_0, \dots, \mathbf{v}_{|V|-1}),\ u_i, \mathbf{v}_j \in \mathbb{R}^d$

Derivative w.r.t. center word vector parameters:

$$\begin{aligned}\frac{\partial \log P(w_o \mid w_c)}{\partial \mathbf{v}_c} &= \mathbf{u}_o - \frac{\sum_{j\in\mathcal{V}} \exp(\mathbf{u}_j^\top \mathbf{v}_c)\mathbf{u}_j}{\sum_{i\in\mathcal{V}} \exp(\mathbf{u}_i^\top \mathbf{v}_c)}\\ &= \mathbf{u}_o - \sum_{j\in\mathcal{V}} \left(\frac{\exp(\mathbf{u}_j^\top \mathbf{v}_c)}{\sum_{i\in\mathcal{V}} \exp(\mathbf{u}_i^\top \mathbf{v}_c)}\right)\mathbf{u}_j\\ &= \mathbf{u}_o - \underbrace{\sum_{j\in\mathcal{V}} P(w_j \mid w_c)\mathbf{u}_j}_{}.\end{aligned}$$

Exercise: derivative w.r.t. *context word vectors*?

Average over all context vectors, weighted by their respective probability

Vocabulary $V = \{0, 1, \dots, |V| - 1\}$

$\mathbf{v}_c \coloneqq$ *center word with index* $c \in V$
$\boldsymbol{u}_i \coloneqq$ *context word with index* $i \in V$

Gradient Descent

Idea

Update equation

$$\theta^{new} = \theta^{old} - \alpha \nabla_{\theta^{old}} L(\theta)$$

Gradient Descent

Idea

Update equation

$$\theta^{new} = \theta^{old} - \alpha \underbrace{\nabla_{\theta^{old}} L(\theta)}_{\text{Gradient of loss function}}$$

Gradient Descent

Idea

Update equation

Step size or *learning rate*

$$\theta^{new} = \theta^{old} - \alpha \underbrace{\nabla_{\theta^{old}} L(\theta)}_{\text{Gradient of loss function}}$$

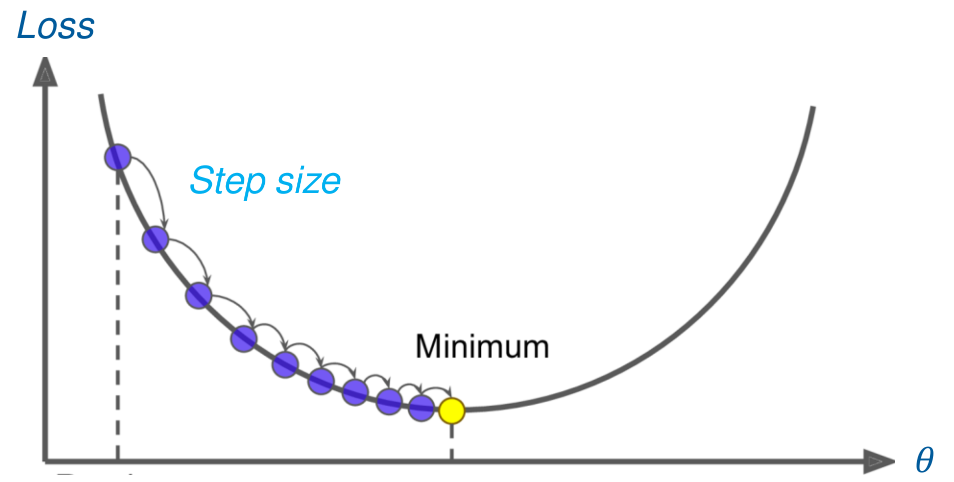
Gradient Descent

Idea

Update equation

Step size or *learning rate*

$$\theta^{new} = \theta^{old} - \alpha \underbrace{\nabla_{\theta^{old}} L(\theta)}_{\text{Gradient of loss function}}$$



$$\theta = \begin{bmatrix} v_{aardvark} \\ v_a \\ \vdots \\ v_{zebra} \\ u_{aardvark} \\ u_a \\ \vdots \\ u_{zebra} \end{bmatrix}$$

Word2Vec: Skip Gram Summary

Idea

Problem: $L(\theta)$ is a function where softmax is computed over all words (and corresponding windows) in vocabulary

Further Reading: https://d2l.ai/chapter_natural-language-processing-pretraining/approx-training.html

Word2Vec: Skip Gram Summary

Idea

Problem: $L(\theta)$ is a function where softmax is computed over all words (and corresponding windows) in vocabulary

$$P(w_o \mid w_c) = \frac{\exp(\mathbf{u}_o^\top \mathbf{v}_c)}{\sum_{i \in \mathcal{V}} \exp(\mathbf{u}_i^\top \mathbf{v}_c)},$$

Further Reading: https://d2l.ai/chapter_natural-language-processing-pretraining/approx-training.html

Word2Vec: Skip Gram Summary

Idea

Problem: $L(\theta)$ is a function where softmax is computed over all words (and corresponding windows) in vocabulary

$$P(w_o \mid w_c) = \frac{\exp(\mathbf{u}_o^\top \mathbf{v}_c)}{\sum_{i \in \mathcal{V}} \exp(\mathbf{u}_i^\top \mathbf{v}_c)},$$

Computationally expensive if vocabulary V is large

… in practice: V **is** very large!

Solution:

- Negative Sampling
- Hierarchical Sampling

Further Reading: https://d2l.ai/chapter_natural-language-processing-pretraining/approx-training.html

Objective Function

`Ein Elektroauto ist ein Automobil mit elektrischem Antrieb.`

Objective Function := $\max \prod_{\text{center}} \prod_{\text{context}} P(\text{context} \mid \text{center})$

Softmax:

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \text{ for } i = 1, \dots, K \text{ and } \mathbf{z} = (z_1, \dots, z_K) \in \mathbb{R}^K$$

Objective Function

`Ein Elektroauto ist ein Automobil mit elektrischem Antrieb.`

Objective Function := $\max \prod_{\text{center}} \prod_{\text{context}} P(\text{context} \mid \text{center})$

$$\min\left(-\frac{1}{N}\sum_{\text{center}}\sum_{\text{context}} \log(P(\text{context} \mid \text{center}))\right)$$:= Commonly used Objective Function

Softmax:

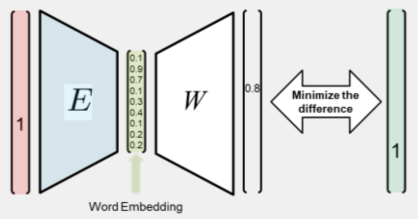
$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \text{ for } i = 1, \ldots, K \text{ and } \mathbf{z} = (z_1, \ldots, z_K) \in \mathbb{R}^K$$

Objective Function

`Ein Elektroauto ist ein Automobil mit elektrischem Antrieb.`

Objective Function := $\max \prod_{\text{center}} \prod_{\text{context}} P(\text{context} \mid \text{center})$

$$\min\left(-\frac{1}{N}\sum_{\text{center}}\sum_{\text{context}} \log(P(\text{context} \mid \text{center}))\right)$$:= Commonly used Objective Function

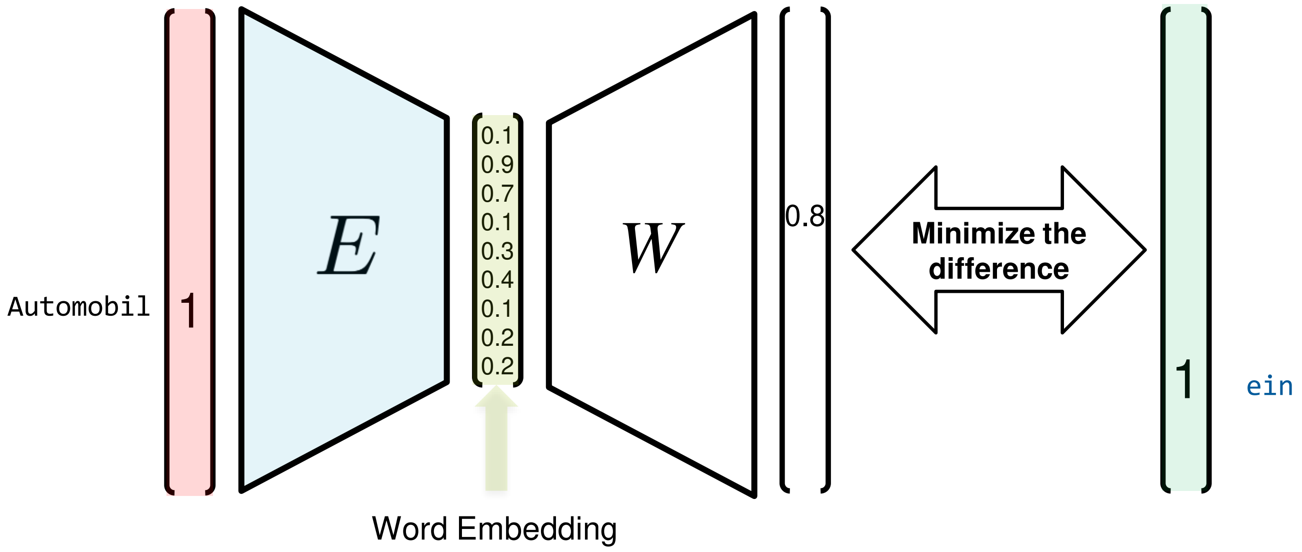


Softmax:

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \text{ for } i = 1, \ldots, K \text{ and } \mathbf{z} = (z_1, \ldots, z_K) \in \mathbb{R}^K$$

Training is computational expensive.

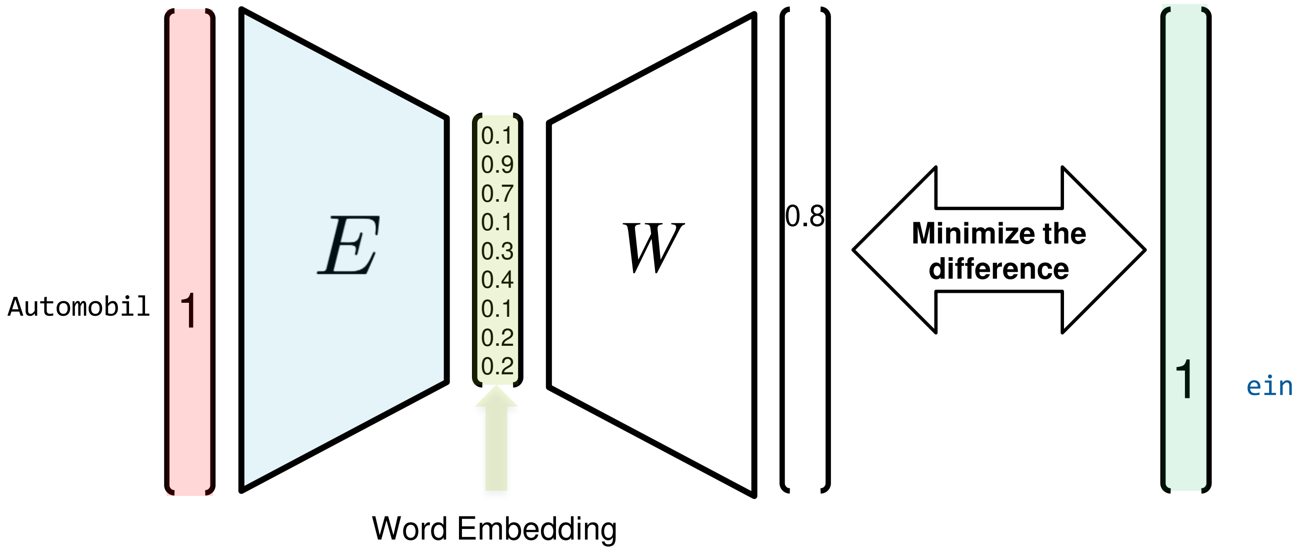
`Ein Elektroauto ist ein Automobil mit elektrischem Antrieb.`



- **Any pair of target/context word**

Training is computational expensive.

`Ein Elektroauto ist ein Automobil mit elektrischem Antrieb.`



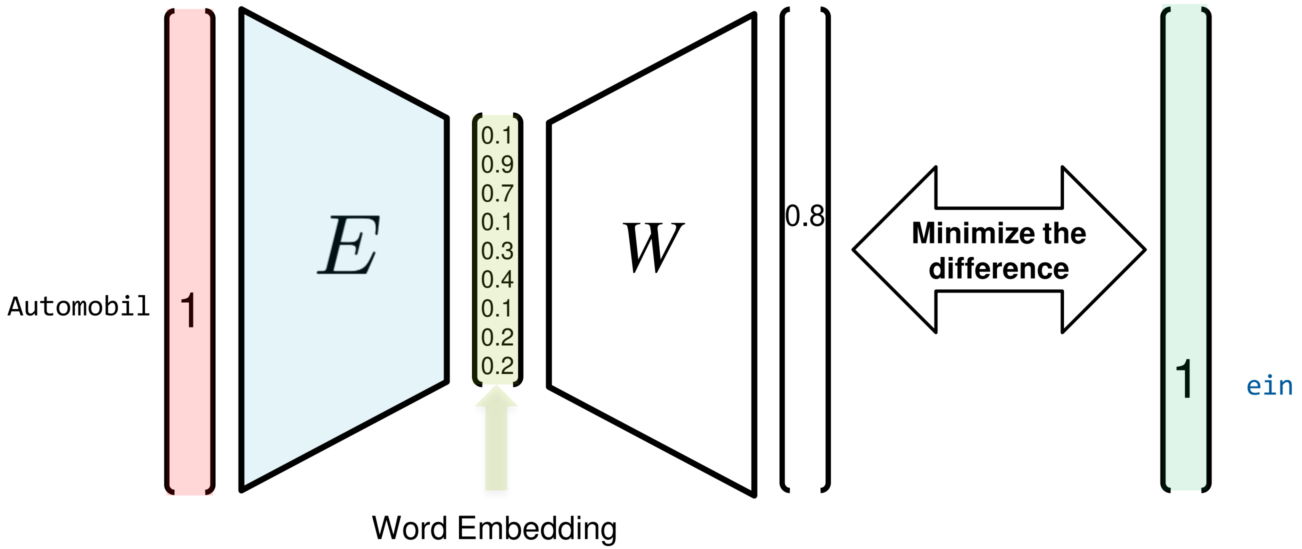
- **Any pair of target/context word**

  Solutions:

  - Sampling

Training is computational expensive.

```
Ein Elektroauto ist ein Automobil mit elektrischem Antrieb.
```



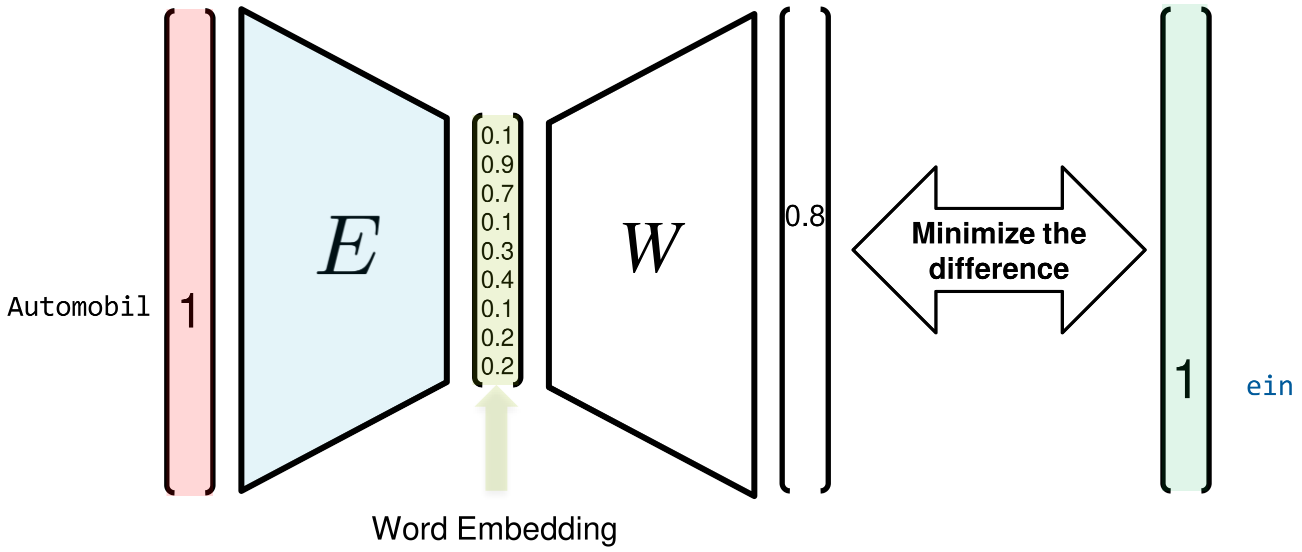
- **Any pair of target/context word**

  Solutions:

  - Sampling
- **Softmax over vocabulary**

Training is computational expensive.

`Ein Elektroauto ist ein Automobil mit elektrischem Antrieb.`



- **Any pair of target/context word**

  Solutions:

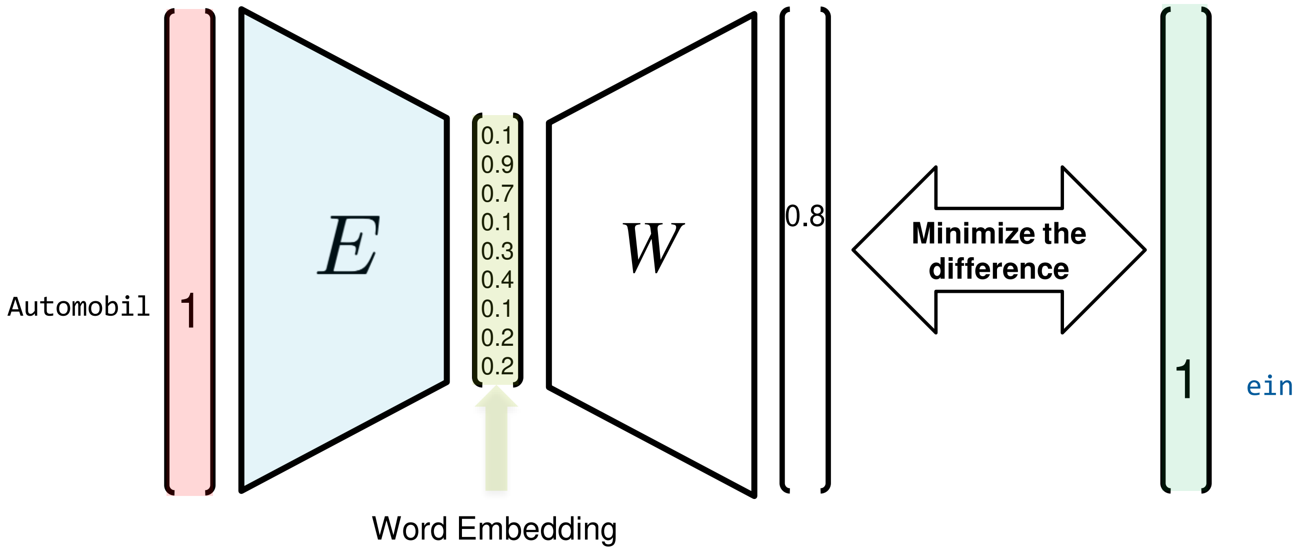
  - Sampling
- **Softmax over vocabulary**

  Solution:

  - Hierarchical Softmax

Training is computational expensive.

`Ein Elektroauto ist ein Automobil mit elektrischem Antrieb.`



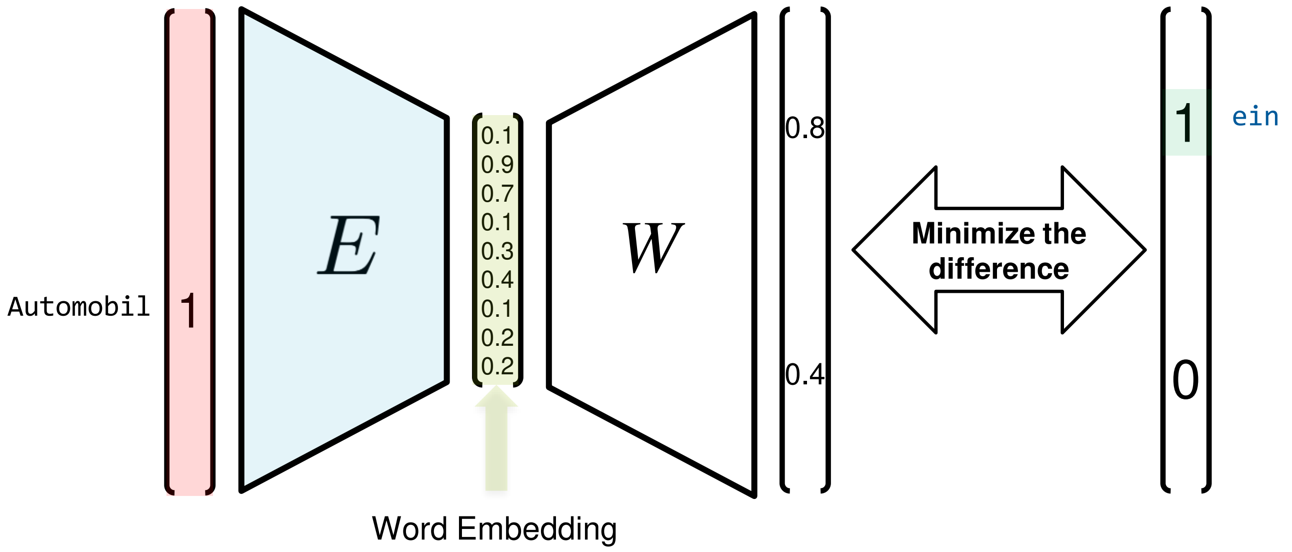
- **Any pair of target/context word**

  Solutions:
  - Sampling
- **Softmax over vocabulary**

  Solution:
  - Hierarchical Softmax
  - Noise Contrastive Estimation
    - Negative Sampling

Training is computational expensive.

`Ein Elektroauto ist ein Automobil mit elektrischem Antrieb.`



Move to binary classification:

- Replace Softmax by Sigmoid
- Train with positive and negative samples

Context	Target		
ein	Automobil	:= 1	Positive Sample
mit	Automobil	:= 1	
Haste	Automobil	:= 0	Negative Sample
oben	Automobil	:= 0	
auf	Automobil	:= 0	

Training is computational expensive.

`Ein Elektroauto ist ein Automobil mit elektrischem Antrieb.`

$P(+|t,c)$

Positive Sample

Context	Target		
ein	Automobil	:= 1	Positive Sample
mit	Automobil	:= 1	

Training is computational expensive.

`Ein Elektroauto ist ein Automobil mit elektrischem Antrieb.`

This is a binary classifier $\left\{ \begin{array}{l} P(+|t,c) \\ \text{Positive Sample} \end{array} \right.$

Context	Target		
ein	Automobil	:= 1	Positive Sample
mit	Automobil	:= 1	

Training is computational expensive. Replace softmax with sigmoid.

`Ein Elektroauto ist ein Automobil mit elektrischem Antrieb.`

This is a binary classifier $\left\{ \begin{array}{l} P(+|t,c) \\ \text{Positive Sample} \end{array} \right.$

$$P(+|t,c) = \frac{1}{1+e^{-t \cdot c}}$$

Context	Target		
ein	Automobil	:= 1	Positive Sample
mit	Automobil	:= 1	

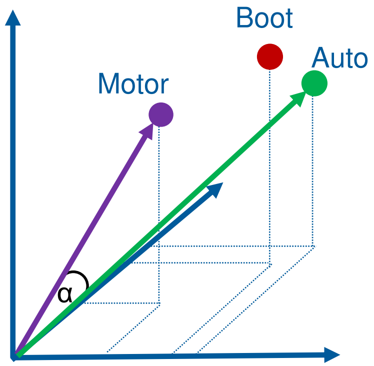
Sigmoid:

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

Training is computational expensive. Replace softmax with sigmoid.

`Ein Elektroauto ist ein Automobil mit elektrischem Antrieb.`

This is a binary classifier $\left\{ \begin{array}{l} P(+|t,c) \\ \text{Positive Sample} \end{array} \right.$



$$P(+|t,c) = \frac{1}{1+e^{-t \cdot c}}$$

$$Similarity(t,c) \approx t \cdot c$$

Context	Target		
ein	Automobil	:= 1	Positive Sample
mit	Automobil	:= 1	

Cosine Similarity:

$$\cos \sphericalangle(\vec{a}, \vec{b}) = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}|\,|\vec{b}|}$$

Training is computational expensive. A (binary) classifier needs “negative samples”.

`Ein Elektroauto ist ein Automobil mit elektrischem Antrieb.`

$P(-|t,c) = 1 - P(+|t,c)$

This is a binary classifier $\left\{ \begin{array}{l} P(+|t,c) \\ \text{Positive Sample} \end{array} \right.$

Negative Sample

`Schinken`

$$P(+|t,c) \; = \; \frac{1}{1+e^{-t \cdot c}}$$

Context	Target		
ein	Automobil	:= 1	Positive Sample
mit	Automobil	:= 1	
Haste	Automobil	:= 0	Negative Sample
oben	Automobil	:= 0	
auf	Automobil	:= 0	

Sigmoid:

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

Training is computational expensive. A (binary) classifier needs “negative samples”.

`Ein Elektroauto ist ein Automobil mit elektrischem Antrieb.`

$$P(-|t,c) = 1 - P(+|t,c)$$

This is a binary classifier $\left\{ \begin{array}{l} P(+|t,c) \\ \text{Positive Sample} \end{array} \right.$

Negative Sample

`Schinken`

$$P(+|t,c) = \frac{1}{1+e^{-t\cdot c}}$$

$$P(-|t,c) = 1 - P(+|t,c) = \frac{e^{-t\cdot c}}{1+e^{-t\cdot c}}$$

Sigmoid:

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

Word Embedding

Training Skip-Gram Word Embedding with Negative Sampling

- Maximize the similarity of the target word, context word pairs (t,c) drawn from the positive examples
- Minimize the similarity of the (t,c) pairs drawn from the negative examples.

`Ein Elektroauto ist ein Automobil mit elektrischem Antrieb.`

$$P(-|t,c) = 1 - P(+|t,c)$$

This is a binary classifier

$P(+|t,c)$

Positive Sample

Negative Sample

`Schinken`

$$P(+|t,c) \;=\; \frac{1}{1+e^{-t\cdot c}} \qquad\qquad P(-|t,c) \;=\; 1 - P(+|t,c) = \frac{e^{-t\cdot c}}{1+e^{-t\cdot c}}$$

$$L(\theta) \;=\; \log\frac{1}{1+e^{-c\cdot t}} + \sum_{i=1}^{k}\log\frac{1}{1+e^{n_i\cdot t}}$$

For k noise samples

Sigmoid:

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

Training is computational expensive.

`Ein Elektroauto ist ein Automobil mit elektrischem Antrieb.`

$P(-|t,c) = 1 - P(+|t,c)$

Remember: We have a context window

$$P(+|t,c_{1:k}) = \prod_{i=1}^{k} \frac{1}{1+e^{-t \cdot c_i}}$$

$$\log P(+|t,c_{1:k}) = \sum_{i=1}^{k} \log \frac{1}{1+e^{-t \cdot c_i}}$$

Independents of context words assumed

This is a binary classifier { $P(+|t,c)$ Positive Sample

Negative Sample

`Schinken`

$$P(+|t,c) = \frac{1}{1+e^{-t \cdot c}}$$

$$P(-|t,c) = 1 - P(+|t,c) = \frac{e^{-t \cdot c}}{1+e^{-t \cdot c}}$$

$$L(\theta) = \log \frac{1}{1+e^{-c \cdot t}} + \sum_{i=1}^{k} \log \frac{1}{1+e^{n_i \cdot t}}$$

For k noise samples

Sigmoid:
$\sigma(x) = \frac{1}{1+e^{-x}}$

Training is computational expensive.

`Ein Elektroauto ist ein Automobil mit elektrischem Antrieb.`

Remember: We have a context window

$$P(+|t,c_{1:k}) = \prod_{i=1}^{k} \frac{1}{1+e^{-t\cdot c_i}}$$

$$\log P(+|t,c_{1:k}) = \sum_{i=1}^{k} \log \frac{1}{1+e^{-t\cdot c_i}}$$

Independents of context words assumed

This is a binary classifier $\left\{ P(+|t,c) \right.$ Positive Sample

$$P(-|t,c) = 1 - P(+|t,c)$$

Negative Sample

`Schinken`

How to select the negative samples from the vocabulary?

$$P(w) = \frac{count(w)}{\sum_{w'} count(w')}$$

Uni-Gram Probabilities

$$P(+|t,c) = \frac{1}{1+e^{-t\cdot c}} \qquad P(-|t,c) = 1 - P(+|t,c) = \frac{e^{-t\cdot c}}{1+e^{-t\cdot c}}$$

$$L(\theta) = \log \frac{1}{1+e^{-c\cdot t}} + \sum_{i=1}^{k} \log \frac{1}{1+e^{n_i\cdot t}}$$

For k noise samples

Sigmoid:

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

Training is computational expensive.

`Ein Elektroauto ist ein Automobil mit elektrischem Antrieb.`

$$P(-|t,c) = 1 - P(+|t,c)$$

Remember: We have a context window

$$P(+|t,c_{1:k}) = \prod_{i=1}^{k} \frac{1}{1+e^{-t\cdot c_i}}$$

$$\log P(+|t,c_{1:k}) = \sum_{i=1}^{k} \log \frac{1}{1+e^{-t\cdot c_i}}$$

Independents of context words assumed

This is a binary classifier

$P(+|t,c)$ Positive Sample

Negative Sample

`Schinken`

How to select the negative samples from the vocabulary?

$$P_\alpha(w) = \frac{count(w)^\alpha}{\sum_{w'} count(w')^\alpha}$$

Weighted Uni-Gram Probabilities

Rare words: $P_\alpha(w) > P(w)$

$$P(+|t,c) = \frac{1}{1+e^{-t\cdot c}} \qquad P(-|t,c) = 1 - P(+|t,c) = \frac{e^{-t\cdot c}}{1+e^{-t\cdot c}}$$

$$L(\theta) = \log \frac{1}{1+e^{-c\cdot t}} + \sum_{i=1}^{k} \log \frac{1}{1+e^{n_i\cdot t}}$$

For k noise samples

Sigmoid:

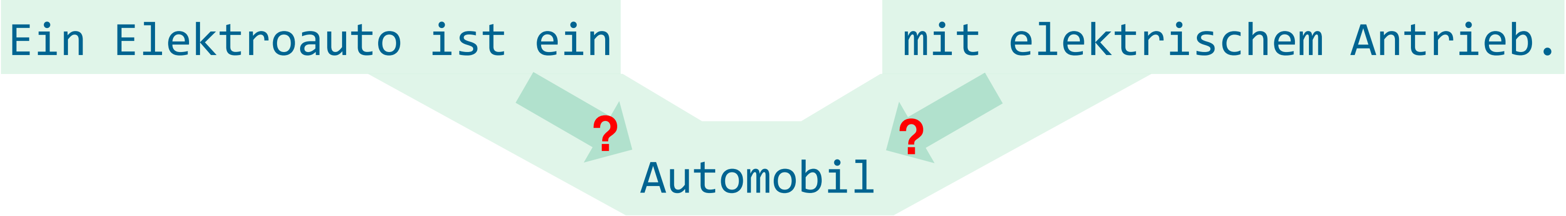
$$\sigma(x) = \frac{1}{1+e^{-x}}$$

https://www.youtube.com/watch?v=vYTihV-9XWE

Word embedding is any of a set of language modeling and feature learning techniques in natural language processing where words from the vocabulary are mapped to vectors of real numbers.

```
Ein Elektroauto ist ein Automobil mit elektrischem Antrieb.
```

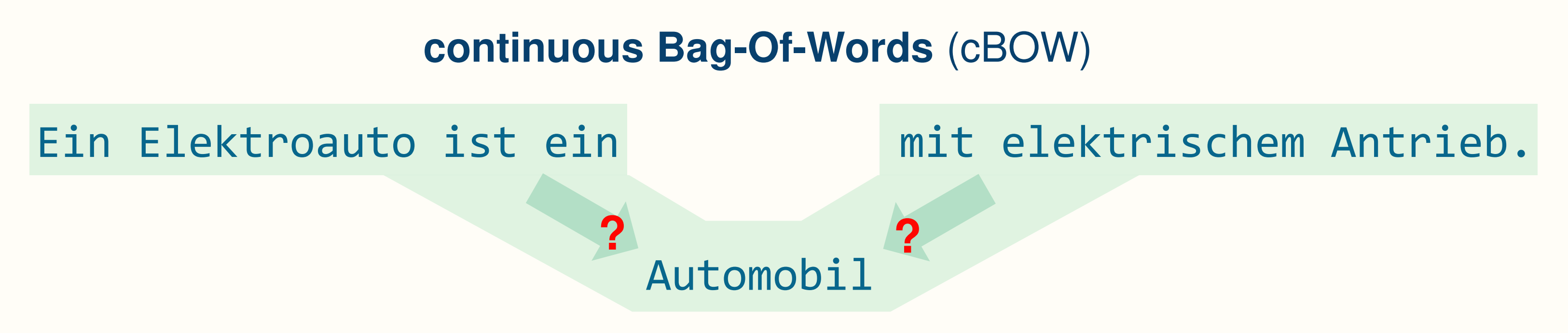
Word embedding is any of a set of language modeling and feature learning techniques in natural language processing where words from the vocabulary are mapped to vectors of real numbers.



Word Embedding

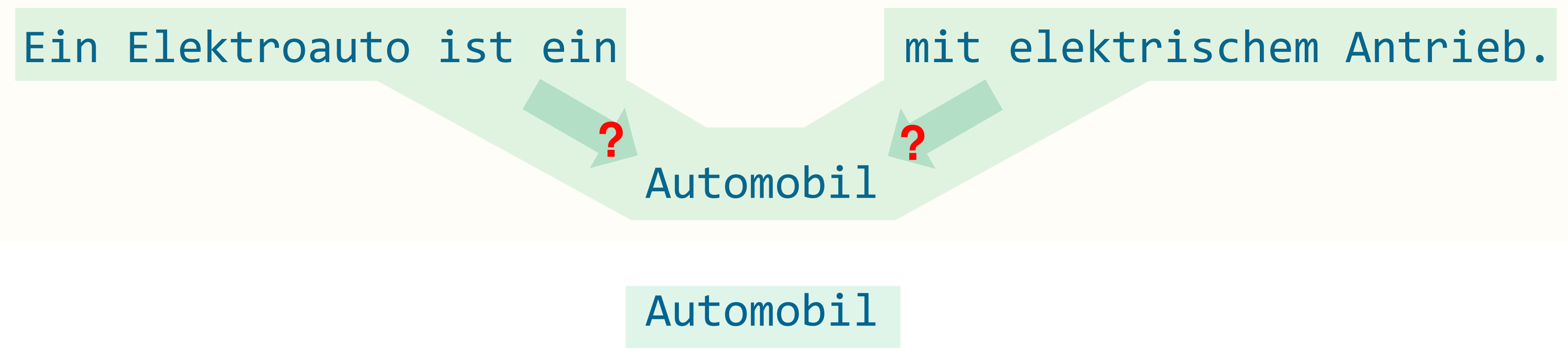
Transform a Word Vector

Word embedding is any of a set of language modeling and feature learning techniques in natural language processing where words from the vocabulary are mapped to vectors of real numbers.

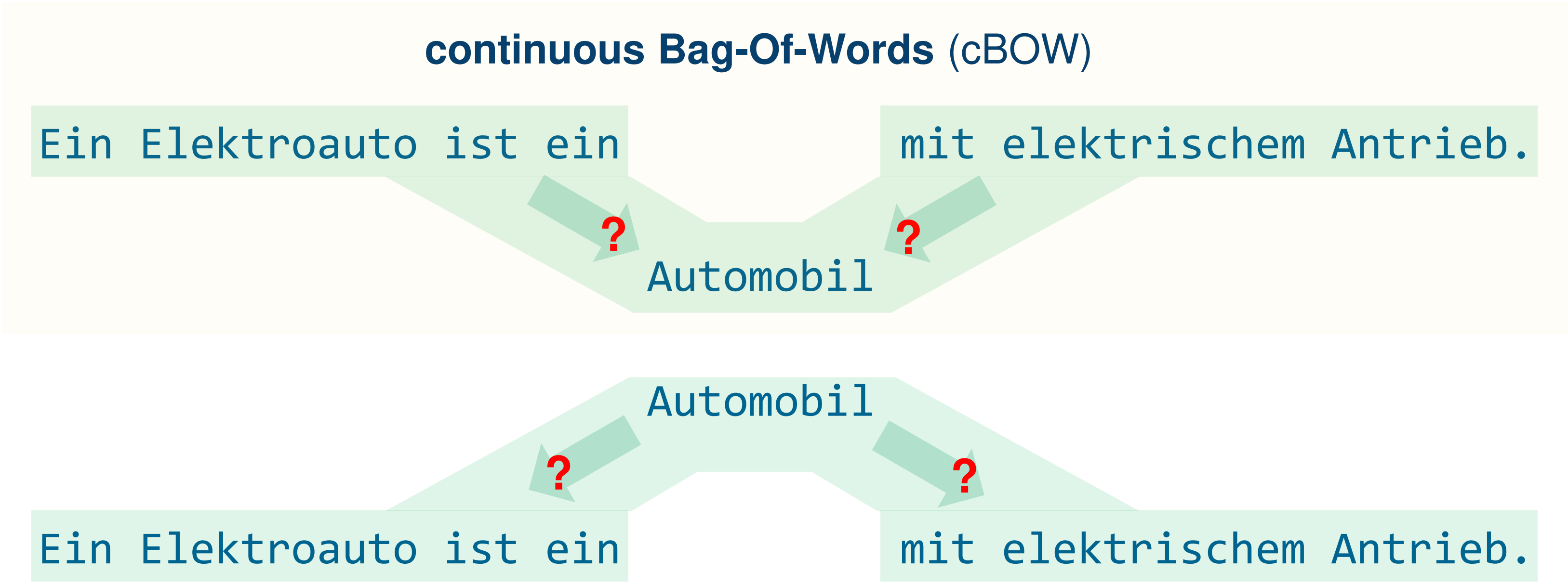


Word embedding is any of a set of language modeling and feature learning techniques in natural language processing where words from the vocabulary are mapped to vectors of real numbers.

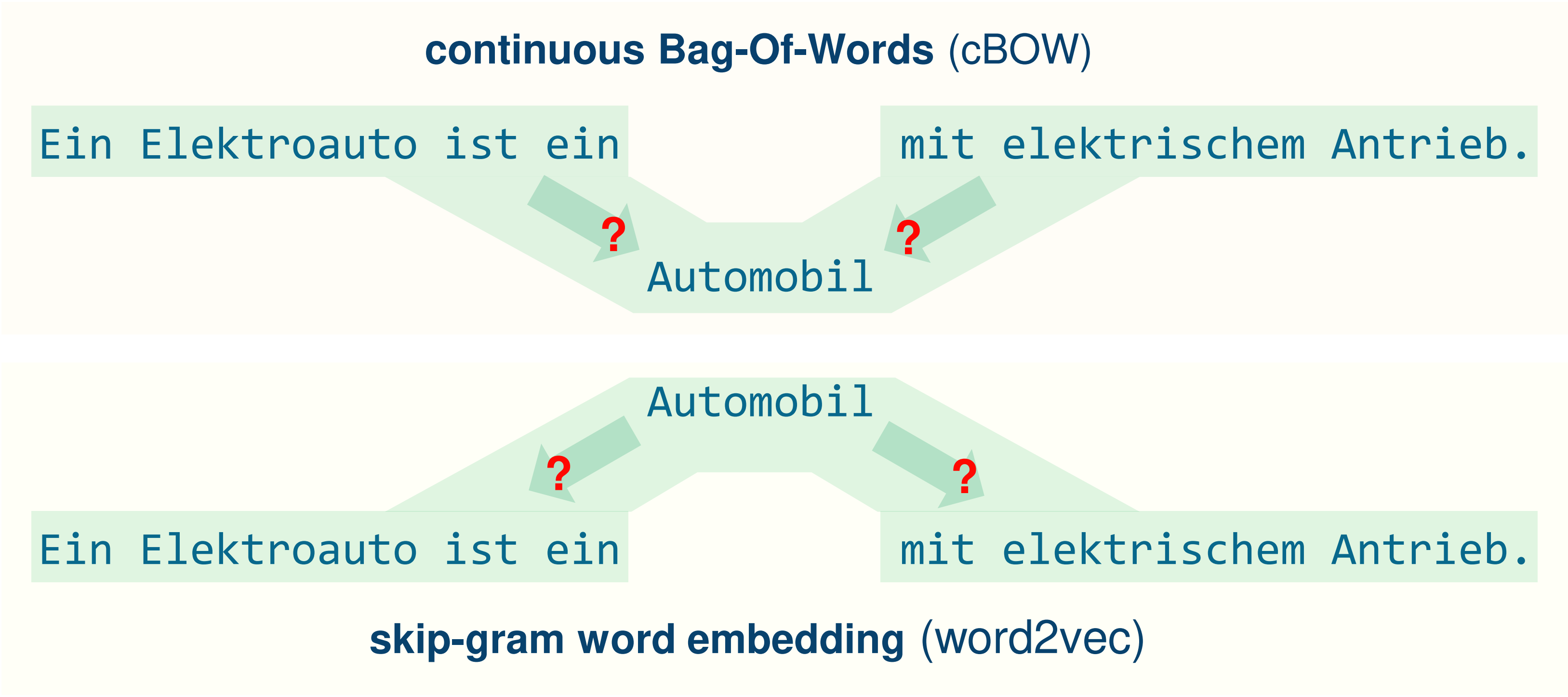
continuous Bag-Of-Words (cBOW)



Word embedding is any of a set of language modeling and feature learning techniques in natural language processing where words from the vocabulary are mapped to vectors of real numbers.



Word embedding is any of a set of language modeling and feature learning techniques in natural language processing where words from the vocabulary are mapped to vectors of real numbers.



The input is always one word of the sentence

`Ein Elektroauto ist ein Automobil mit elektrischem Antrieb.`

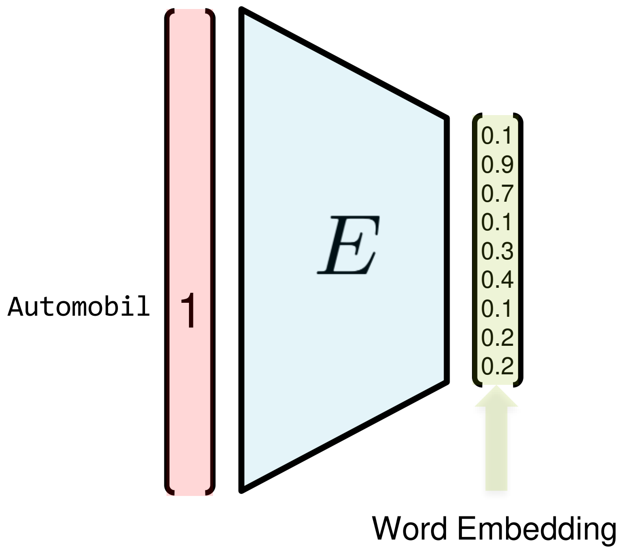
```
for targetword in sentence:
```

$$\begin{array}{rc} & \\ \texttt{Automobil} & \begin{bmatrix} \\ 1 \\ \\ \end{bmatrix} \end{array}$$

$$x_i = E w_i$$

Compute the word embedding of given word

`Ein Elektroauto ist ein Automobil mit elektrischem Antrieb.`

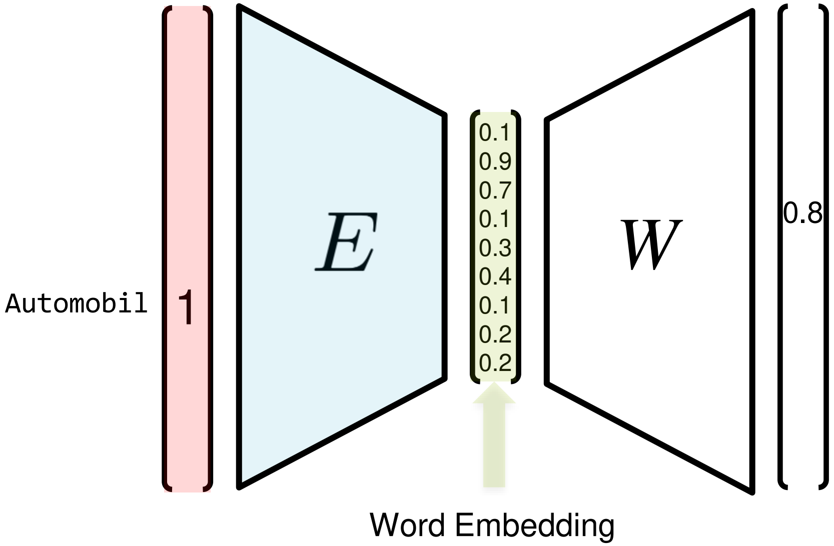



```
for targetword in sentence:
    embedding = matmul(E,targetword)
```

$$x_i = E w_i$$

Make a prediction using the word embedding aka “use the word embedding”

`Ein Elektroauto ist ein Automobil mit elektrischem Antrieb.`



$$= Softmax(Wx_i)$$

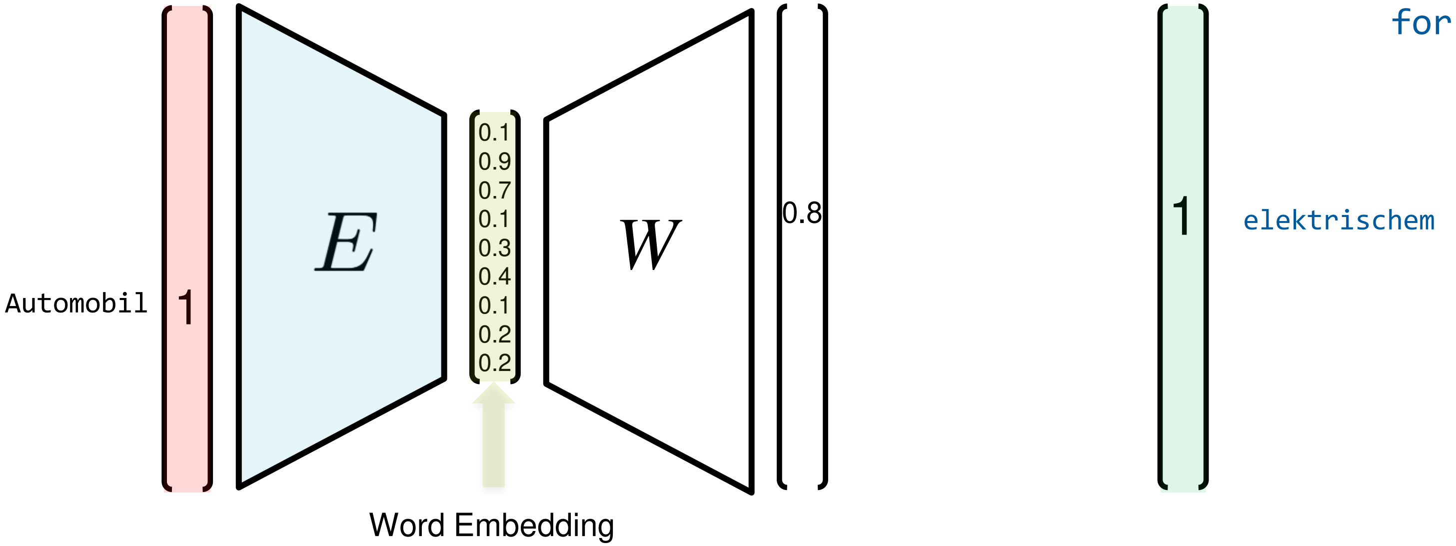
```
for targetword in sentence:

    embedding = matmul(E,targetword)
    tmp = matmul(W, embedding)
    predicted_contextword = softmax(tmp)
```

$$x_i = Ew_i$$

One possible prediction: Predict the context of the word

Context: +/- 2 Words

`Ein Elektroauto ist ein Automobil mit elektrischem Antrieb.`



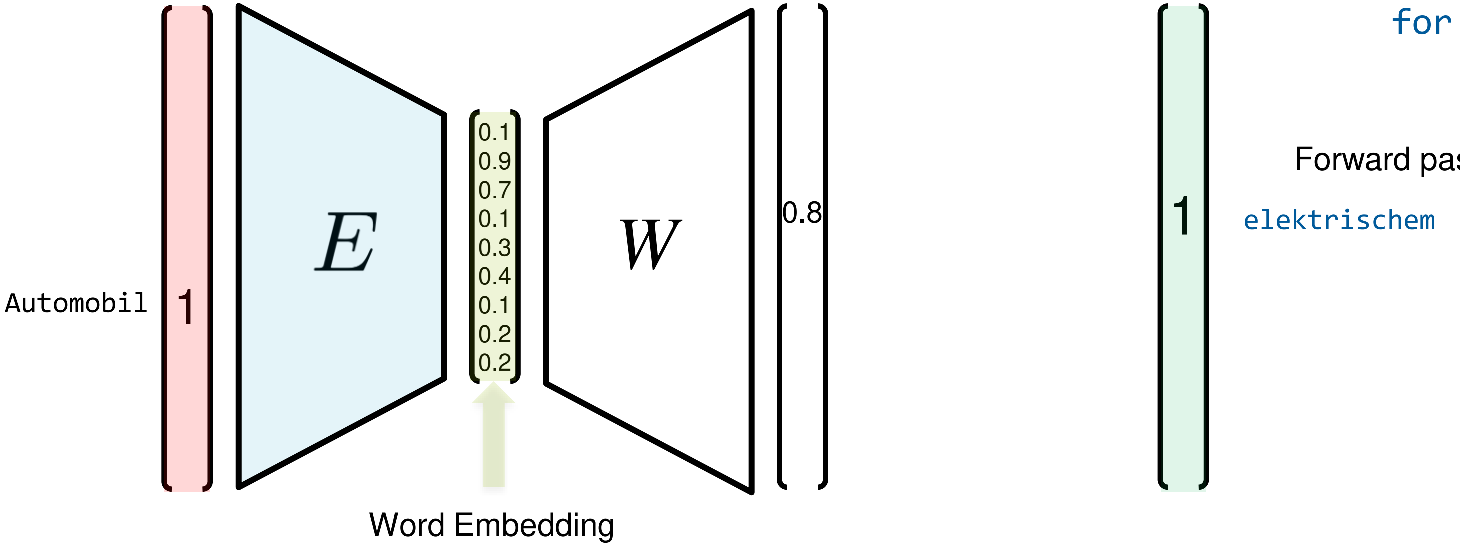

```
for targetword in sentence:
    for contextword arround targetword:
        embedding = matmul(E,targetword)
        tmp = matmul(W, embedding)
        predicted_contextword = softmax(tmp)
```

$$x_i = E w_i$$

One possible prediction: Predict the context of the word

$$\underbrace{\texttt{ist ein Automobil mit elektrischem}}_{\text{Context: +/- 2 Words}}$$

`Ein Elektroauto ist ein Automobil mit elektrischem Antrieb.`



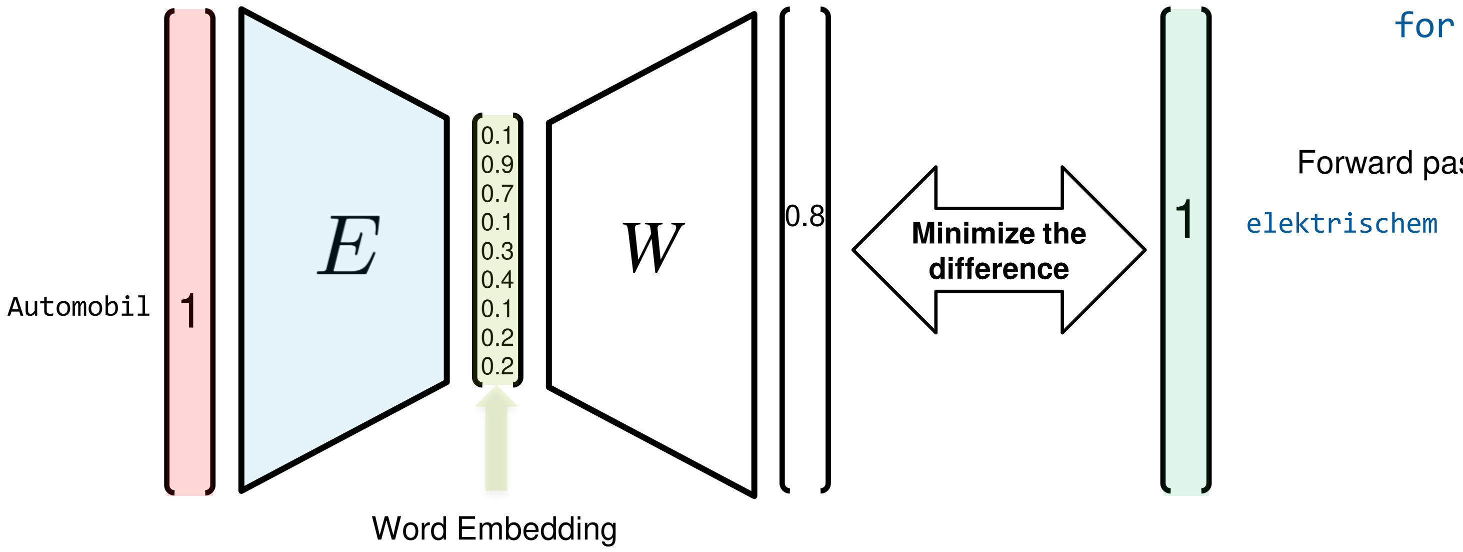

```
for targetword in sentence:
    for contextword arround targetword:
        embedding = matmul(E,targetword)
        tmp = matmul(W, embedding)
        predicted_contextword = softmax(tmp)
```

Forward pass

$$x_i = E w_i$$

Minimize the error between prediction of the context and the real context by updating the network

`Ein Elektroauto ist ein Automobil mit elektrischem Antrieb.`



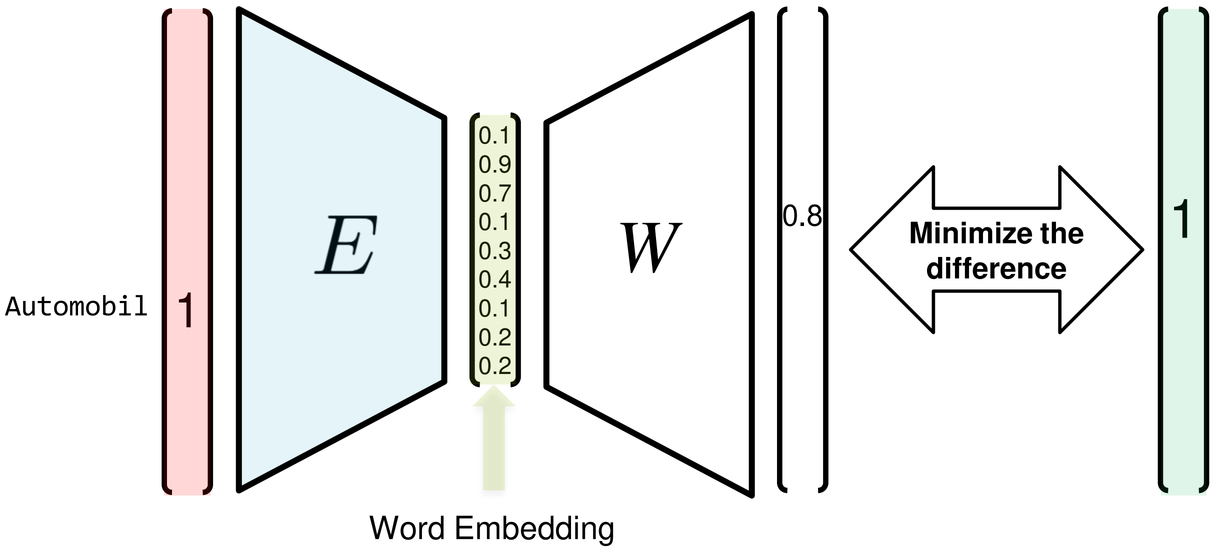
Forward pass

```
for targetword in sentence:
    for contextword arround targetword:
        embedding = matmul(E,targetword)
        tmp = matmul(W, embedding)
        predicted_contextword = softmax(tmp)
        minimize(predicted_contextword, contextword)
```

$$x_i = E w_i$$

Minimize the error between prediction of the context and the real context by updating the network

`Ein Elektroauto ist ein Automobil mit elektrischem Antrieb.`




```
for targetword in sentence:
    for contextword arround targetword:
        embedding = matmul(E,targetword)
        tmp = matmul(W, embedding)
        predicted_contextword = softmax(tmp)
        minimize(predicted_contextword, contextword)
```

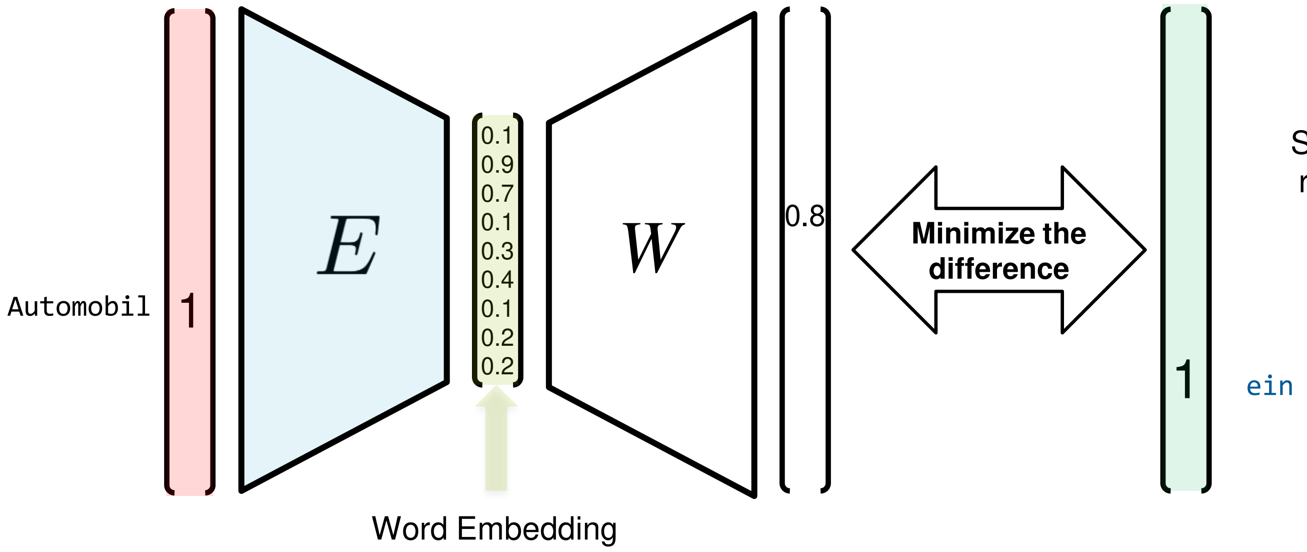
Forward pass

Backward pass
Updates

$$x_i = E w_i$$

Considering the complete context of a word is almost impossible: sample.

`Ein Elektroauto ist ein Automobil mit elektrischem Antrieb.`



Sampling required

```
for targetword in sentence:
    for contextword arround targetword:
        embedding = matmul(E,targetword)
        tmp = matmul(W, embedding)
        predicted_contextword = softmax(tmp)
        minimize(predicted_contextword, contextword)
```

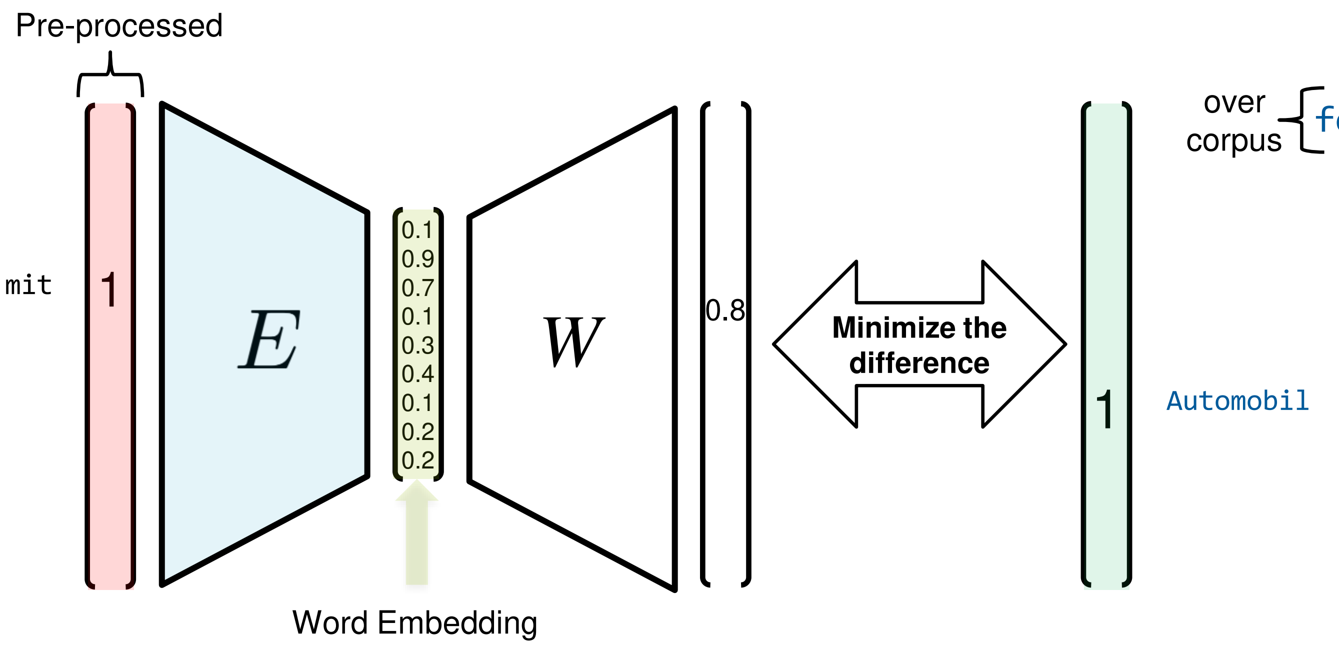
$$x_i = E w_i$$

Word Embedding

Skip-Gram Word Embedding

Repeat the process for any word in the corpus

`Ein Elektroauto ist ein Automobil mit elektrischem Antrieb.`

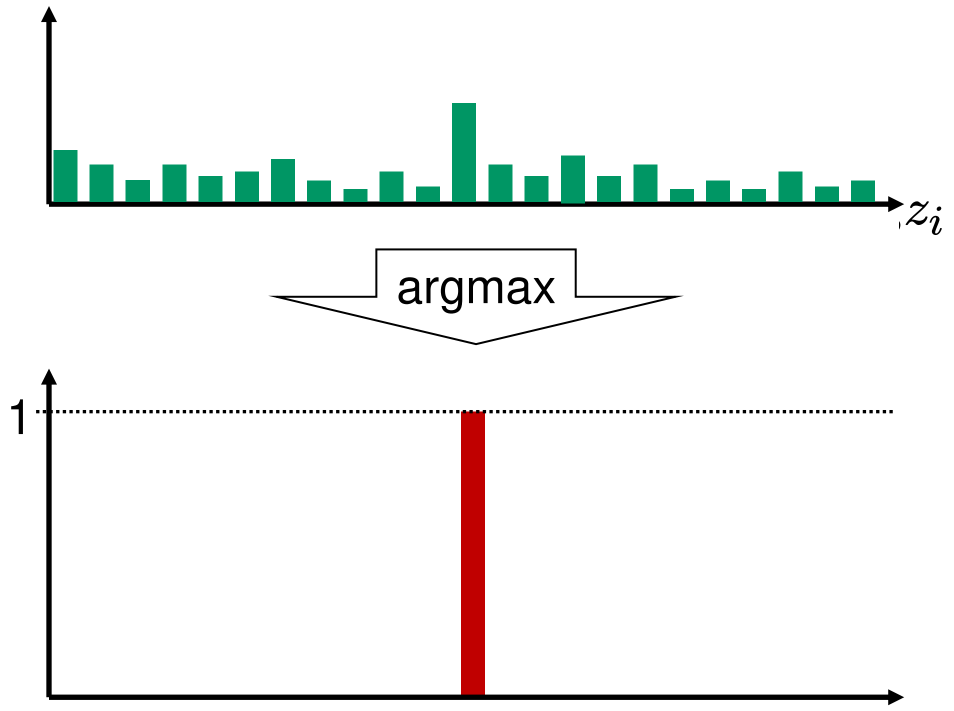


over corpus

```
for targetword in sentence:
    for contextword arround targetword:
        embedding = matmul(E,targetword)
        tmp = matmul(W, embedding)
        predicted_contextword = softmax(tmp)
        minimize(predicted_contextword, contextword)
```

$$x_i = E w_i$$

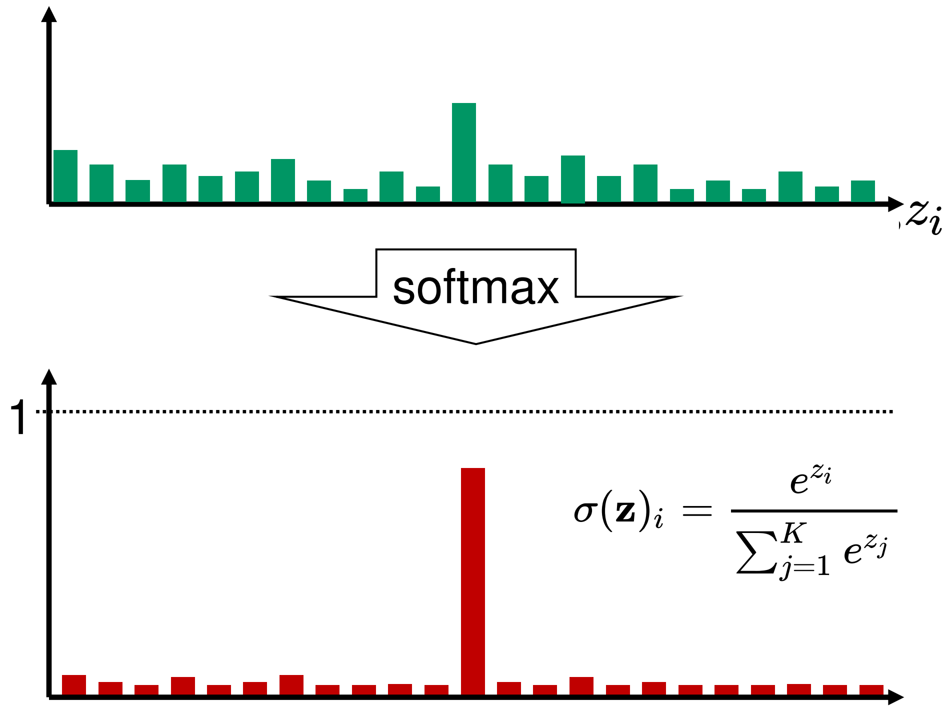
The softmax function is a function that turns a vector of K real values into a vector of K real values that sum to 1.



Where is the maximum?

All values are 0 except the one where the maximum is. This value is 1. Hence, the sum of all entries is 1, too.

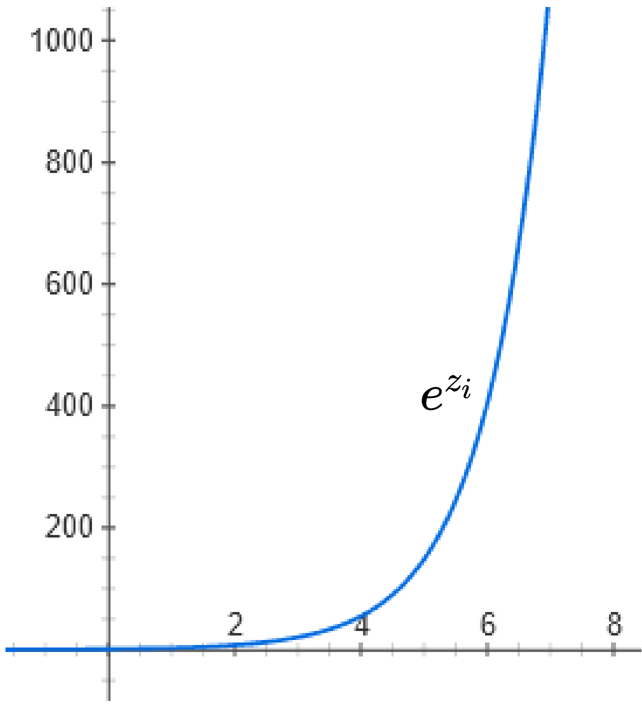
The softmax function is a function that turns a vector of K real values into a vector of K real values that sum to 1.



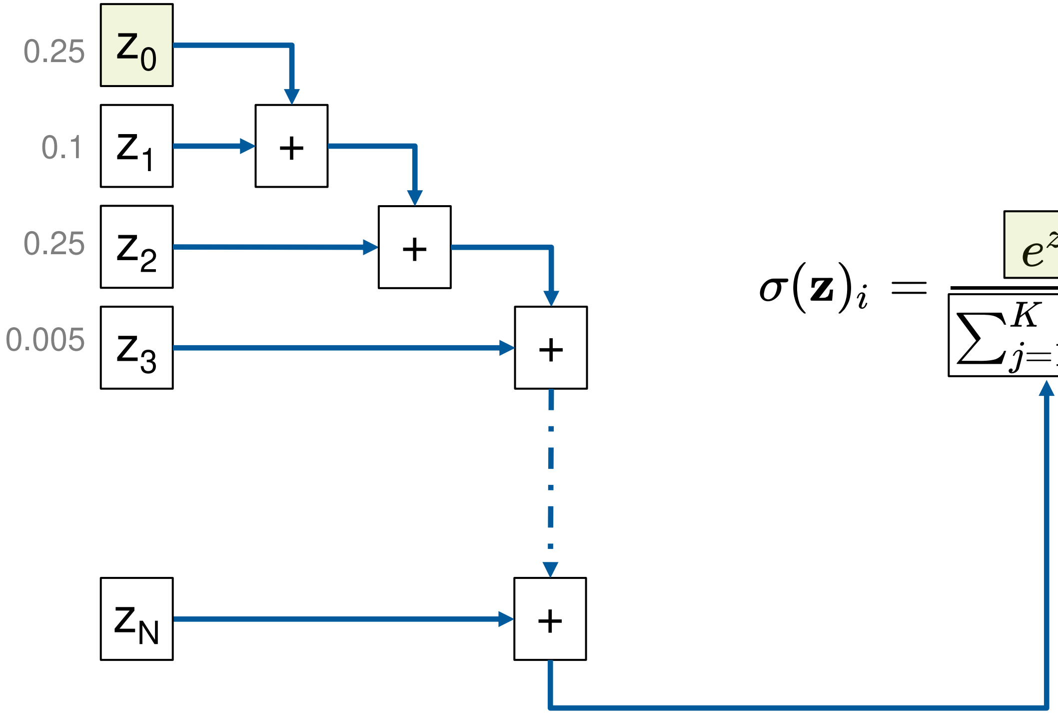
$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

Where is the maximum?

Maximum is significantly larger than all other values. All other values are very close to 0.



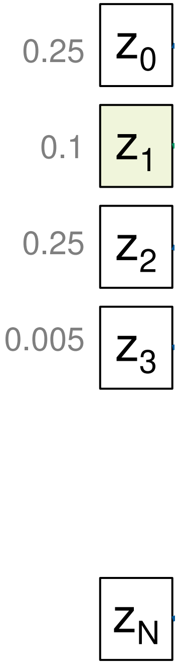
The softmax function is a function that turns a vector of K real values into a vector of K real values that sum to 1.



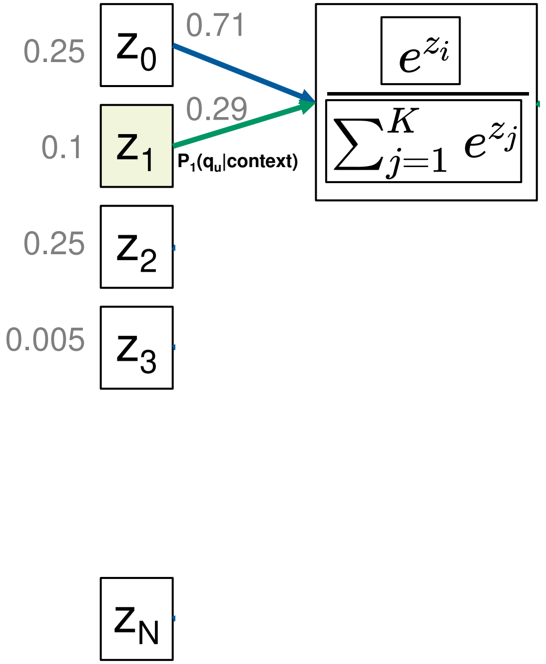
$$\sigma(\mathbf{z})_i = \frac{\boxed{e^{z_i}}}{\boxed{\sum_{j=1}^{K} e^{z_j}}} \text{ for } i = 1, \ldots, K \text{ and } \mathbf{z} = (z_1, \ldots, z_K) \in \mathbb{R}^K$$

https://en.wikipedia.org/wiki/Softmax_function

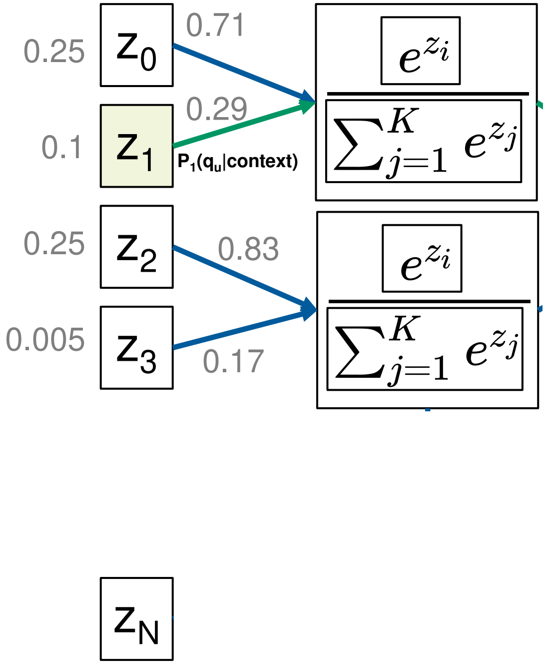
An approximation inspired by binary trees that replaces the flat softmax layer with a hierarchical layer that has the words as leaves.



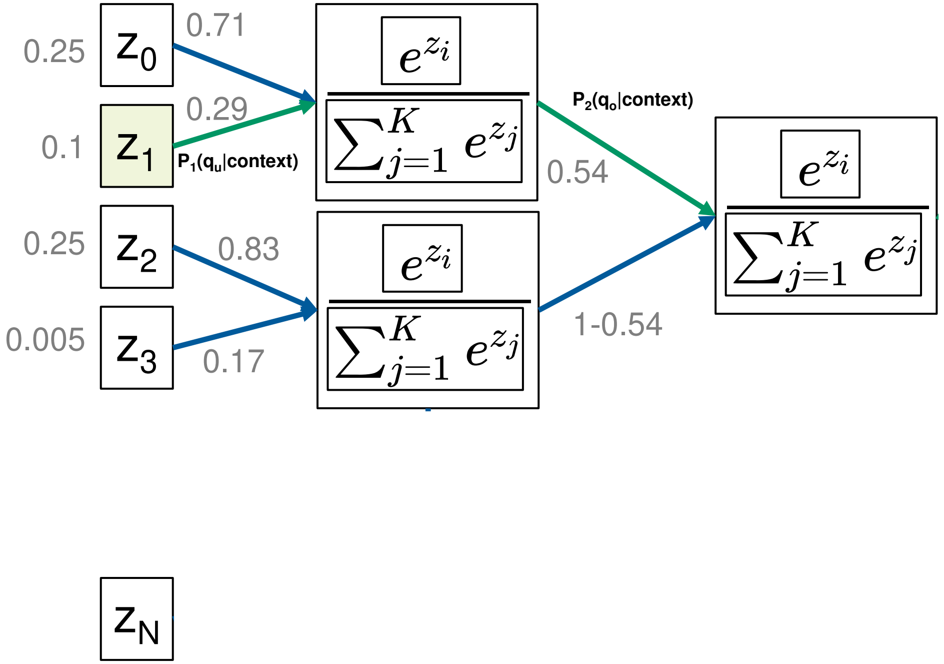
An approximation inspired by binary trees that replaces the flat softmax layer with a hierarchical layer that has the words as leaves.



An approximation inspired by binary trees that replaces the flat softmax layer with a hierarchical layer that has the words as leaves.



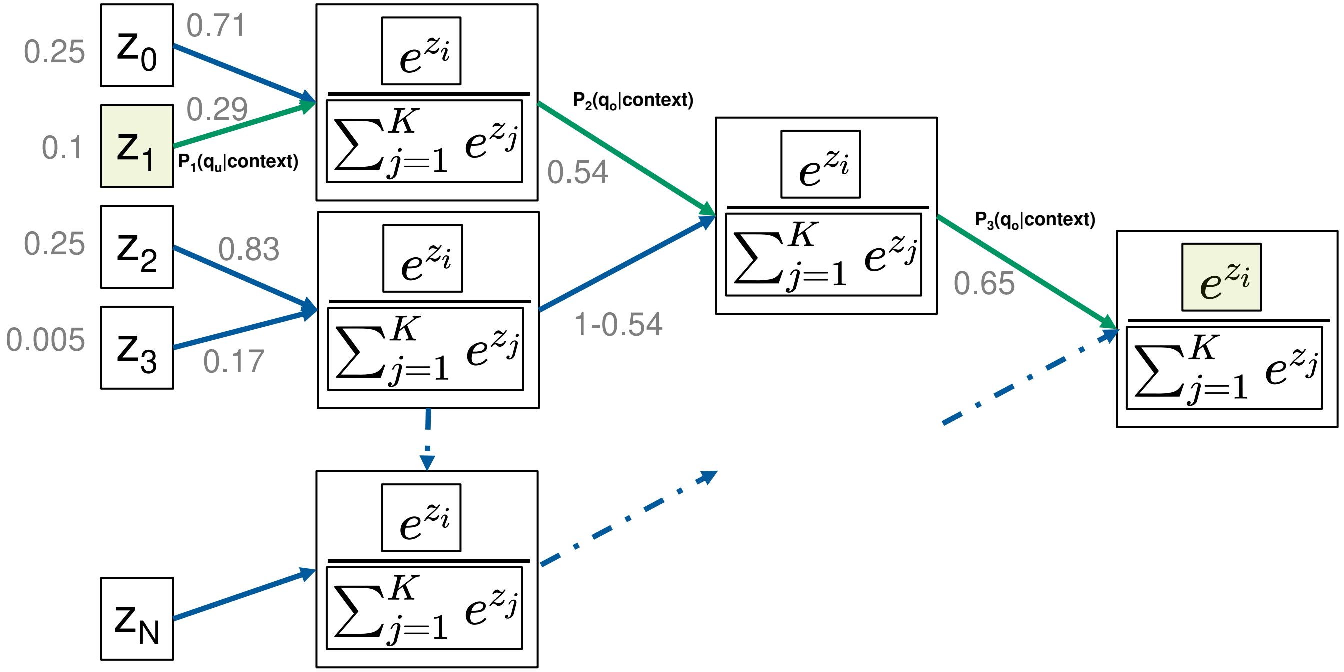
An approximation inspired by binary trees that replaces the flat softmax layer with a hierarchical layer that has the words as leaves.



Softmax

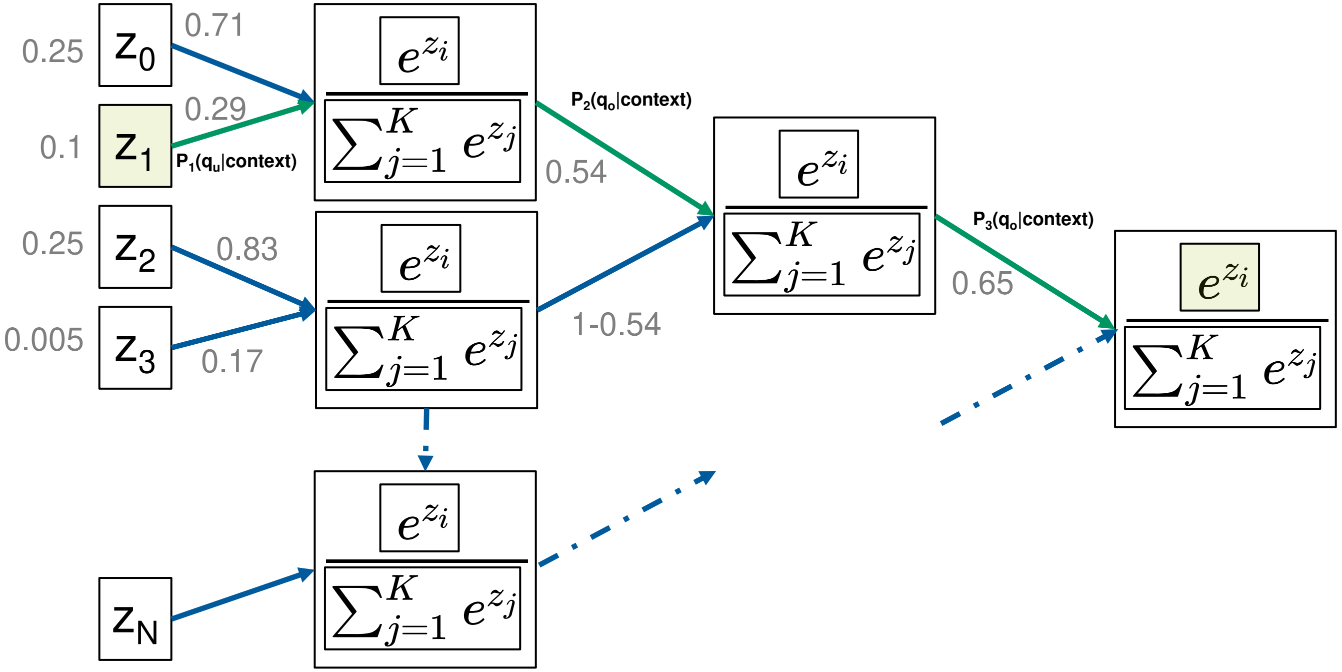
Hierarchical Softmax

An approximation inspired by binary trees that replaces the flat softmax layer with a hierarchical layer that has the words as leaves.



$$P(w|context) = P_1(q_u|context) \cdot P_2(q_o|context) \cdot P_3(q_o|context)$$

An approximation inspired by binary trees that replaces the flat softmax layer with a hierarchical layer that has the words as leaves.

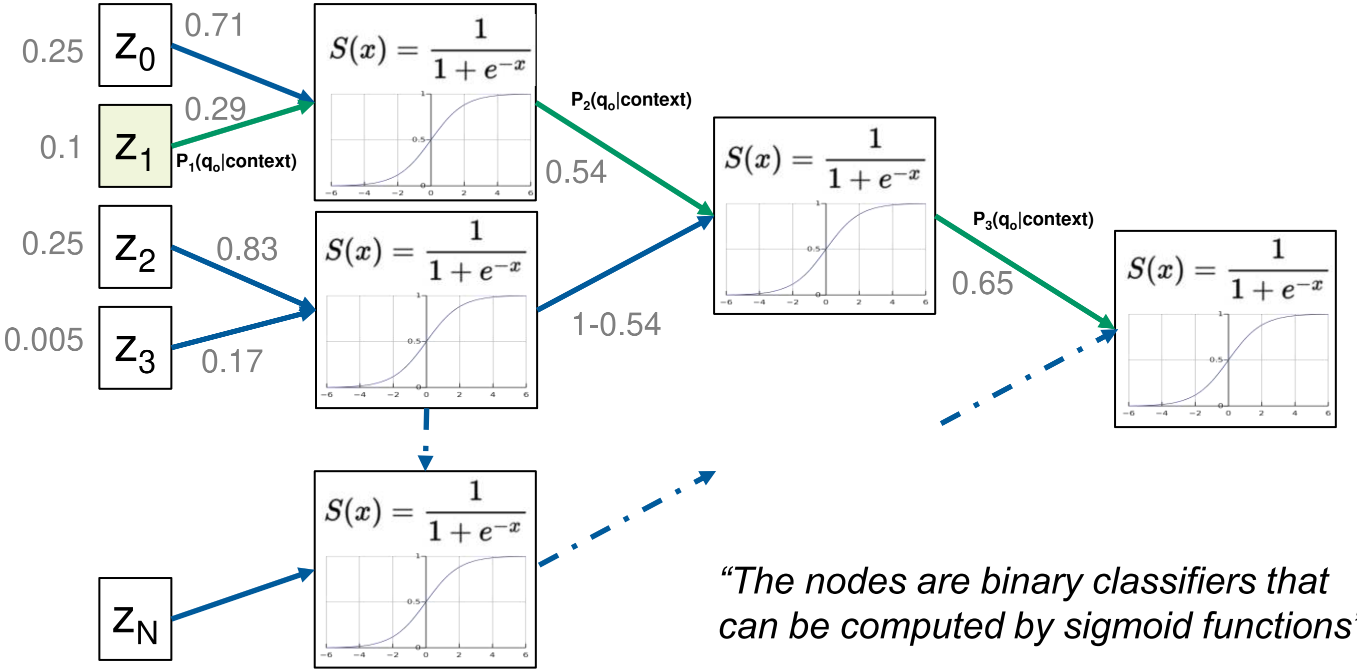


"The probability of a word w given a context, is encoded as path from root to word w."

Complexity reduces to log(|V|) for binary trees

$$P(w|context) = P_1(q_u|context) \cdot P_2(q_o|context) \cdot P_3(q_o|context)$$

An approximation inspired by binary trees that replaces the flat softmax layer with a hierarchical layer that has the words as leaves.



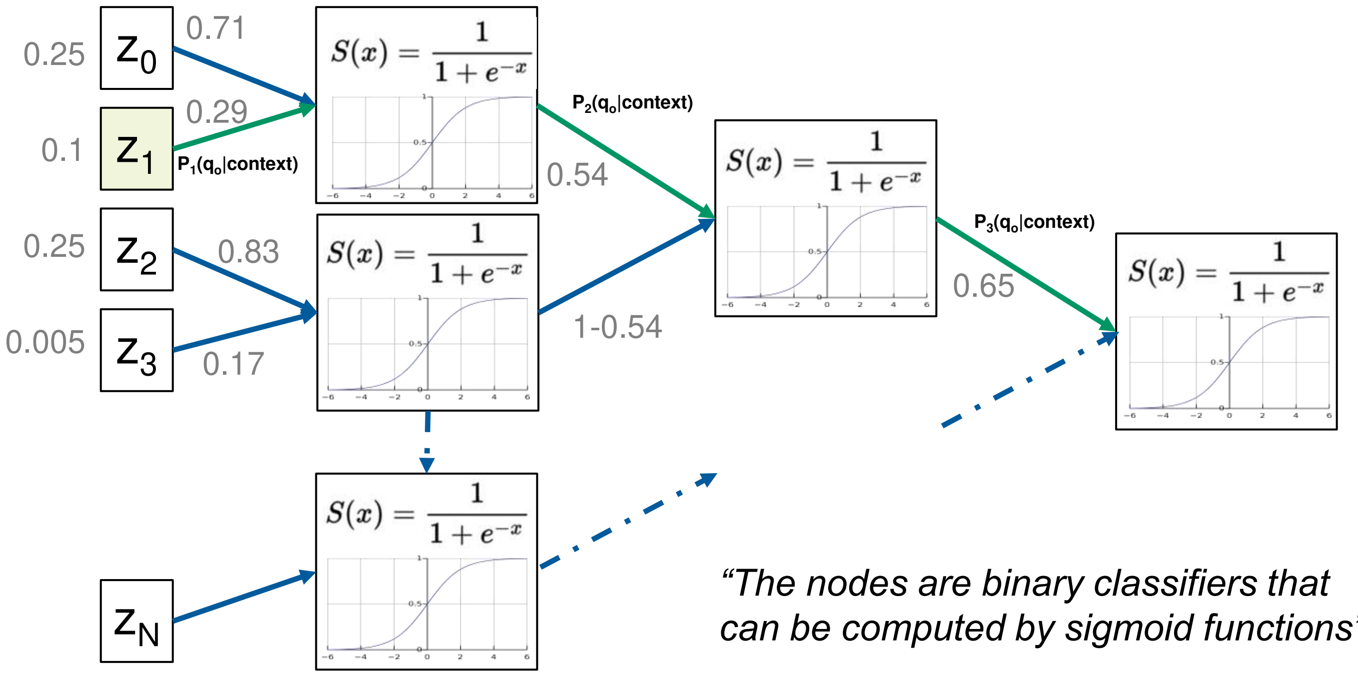
"The probability of a word w given a context, is encoded as path from root to word w."

Complexity reduces to log(|V|) for binary trees

$$P(w|context) = P_1(q_u|context) \cdot P_2(q_o|context) \cdot P_3(q_o|context)$$

"The nodes are binary classifiers that can be computed by sigmoid functions"

An approximation inspired by binary trees that replaces the flat softmax layer with a hierarchical layer that has the words as leaves.



"The nodes are binary classifiers that can be computed by sigmoid functions"

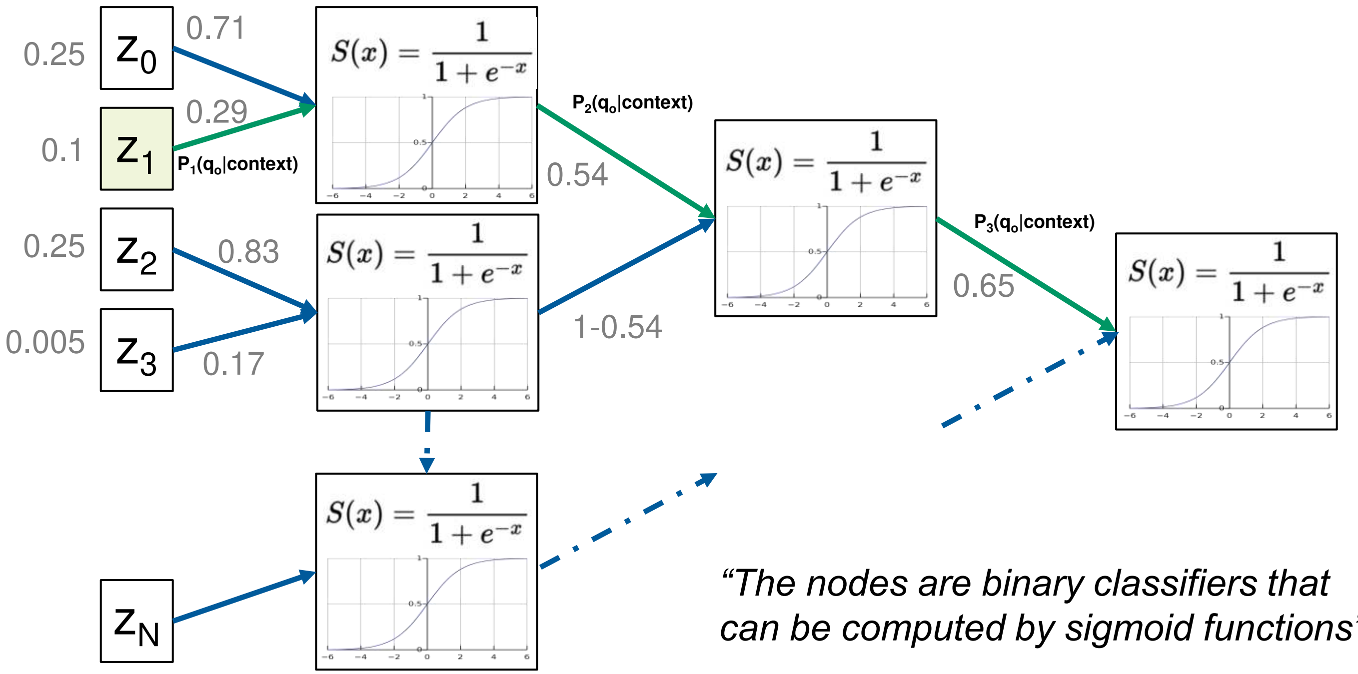
"The probability of a word w given a context, is encoded as path from root to word w."

Complexity reduces to log(|V|) for binary trees

What about none binary trees?

$$P(w|context) = P_1(q_u|context) \cdot P_2(q_o|context) \cdot P_3(q_o|context)$$

An approximation inspired by binary trees that replaces the flat softmax layer with a hierarchical layer that has the words as leaves.



"The nodes are binary classifiers that can be computed by sigmoid functions"

"The probability of a word w given a context, is encoded as path from root to word w."

Complexity reduces to log(|V|) for binary trees

$$P(w|context) = P_1(q_u|context) \cdot P_2(q_o|context) \cdot P_3(q_o|context)$$

What about none binary trees?

Still in research, e.g., wordnet is in discussion…

https://ruder.io/word-embeddings-softmax/index.html#hierarchicalsoftmax
http://building-babylon.net/2017/08/01/hierarchical-softmax/

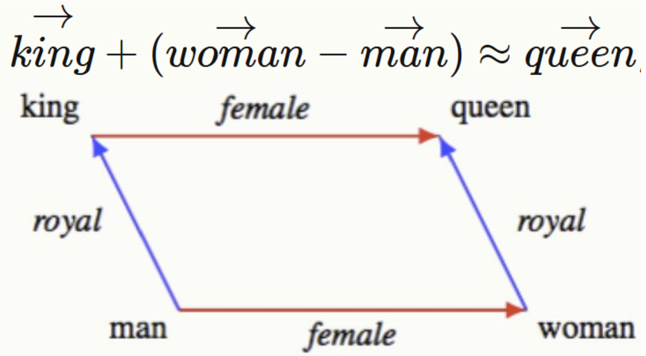
Word Embedding

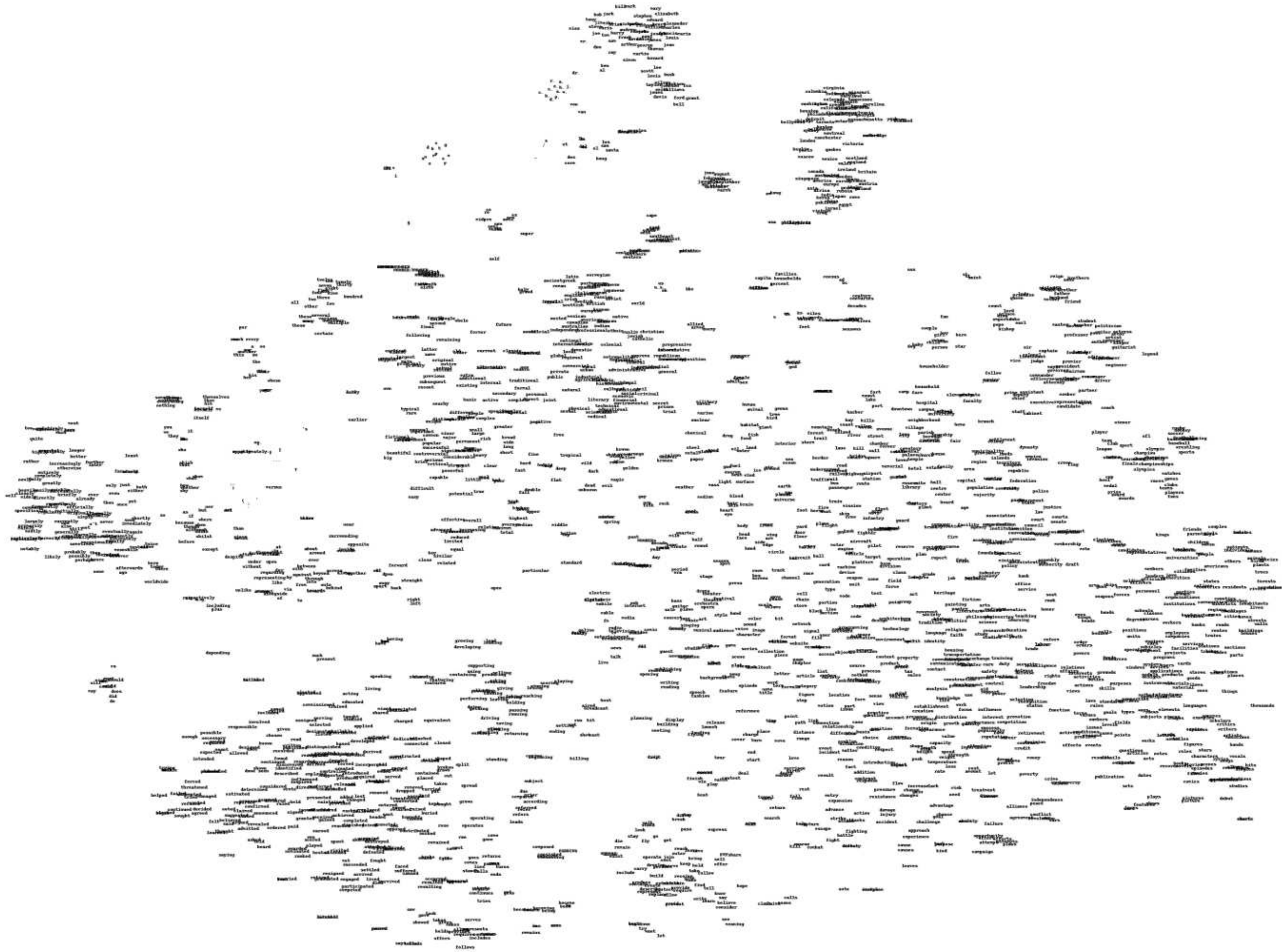
t-distributed stochastic neighbor embedding (T-SNE)

A technique for dimensionality reduction that is particularly well suited for the visualization of high-dimensional datasets.

Online version: https://projector.tensorflow.org/
Examples: https://lvdmaaten.github.io/tsne/
How To: https://distill.pub/2016/misread-tsne/

Be careful with such conclusions, there are million of words!





Word Embedding

Proposals

Embedding from a space with many dimensions per word to a continuous vector space with a much lower dimension.

Generating a word embedding include neural networks, dimensionality reduction on word co-occurrence matrices, probabilistic models, explainable knowledge base method, and explicit representation in terms of the context in which words appear.

(Classic) Word Embedding:

- Word2Vec: cBOW, Skip-Gram
- SGNS (skipgram with negative sampling) can be formalizes as:
- GloVe (:= Global Vectors)
  - trained from global word-word co-occurrence statistics
  - https://nlp.stanford.edu/projects/glove/

$$\mathrm{SGNS} : \langle \vec{x}, \vec{y}_c \rangle = \mathrm{PMI}(x, y) - \log k$$

$$\mathrm{GloVe} : \langle \vec{x}, \vec{y}_c \rangle = \log X_{x,y} - b_x - b_y$$

https://kawine.github.io/blog/nlp/2019/06/21/word-analogies.html

Word Embedding

Proposals

Embedding from a space with many dimensions per word to a continuous vector space with a much lower dimension.

Generating a word embedding include neural networks, dimensionality reduction on word co-occurrence matrices, probabilistic models, explainable knowledge base method, and explicit representation in terms of the context in which words appear.

(Classic) Word Embedding:

- Word2Vec: cBOW, Skip-Gram
- SGNS (skipgram with negative sampling) can be formalizes as: $\text{SGNS}: \langle \vec{x}, \vec{y}_c \rangle = \text{PMI}(x,y) - \log k$
- GloVe (:= Global Vectors) $\text{GloVe}: \langle \vec{x}, \vec{y}_c \rangle = \log X_{x,y} - b_x - b_y$
  - trained from global word-word co-occurrence statistics
  - https://nlp.stanford.edu/projects/glove/

Context Sensitive Word Embedding (The representation for each word depends on the entire context in which it is used.)

- ELMo https://allennlp.org/elmo

https://kawine.github.io/blog/nlp/2019/06/21/word-analogies.html

Word Embedding

Proposals

Embedding from a space with many dimensions per word to a continuous vector space with a much lower dimension.

Generating a word embedding include neural networks, dimensionality reduction on word co-occurrence matrices, probabilistic models, explainable knowledge base method, and explicit representation in terms of the context in which words appear.

(Classic) Word Embedding:

- Word2Vec: cBOW, Skip-Gram
- SGNS (skipgram with negative sampling) can be formalizes as:
- GloVe (:= Global Vectors)
  - trained from global word-word co-occurrence statistics
  - https://nlp.stanford.edu/projects/glove/

$$\mathrm{SGNS} : \langle \vec{x}, \vec{y}_c \rangle = \mathrm{PMI}(x, y) - \log k$$

$$\mathrm{GloVe} : \langle \vec{x}, \vec{y}_c \rangle = \log X_{x,y} - b_x - b_y$$

Context Sensitive Word Embedding (The representation for each word depends on the entire context in which it is used.)

- ELMo https://allennlp.org/elmo

Non-Deterministic:

- T-SNE (t-distributed stochastic neighbor embedding)
  - https://en.wikipedia.org/wiki/T-distributed_stochastic_neighbor_embedding + see exercises

https://kawine.github.io/blog/nlp/2019/06/21/word-analogies.html