Hochschule Offenburg
offenburg.university

Text Analysis

With Embeddings

Outline

Motivation

Text Features and Embeddings

Word2Vec

Doc2Vec

BERT

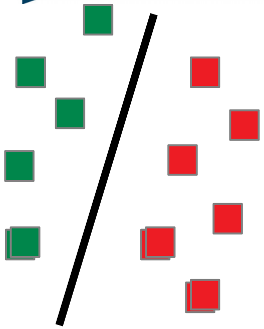
Image by: displayr.com

Motivation

Recall the importance of the feature extraction

In our supervised coffee cup classification:

Function *f* operates in feature space



(X,y)

Feature extraction

$$\begin{pmatrix} 2.06 \\ 1.76 \\ 0.06 \\ -1.7 \\ .. \\ 1.6 \end{pmatrix} \longrightarrow \boxed{\textbf{ML}} \longrightarrow y': \{-1,1\}$$

Feature space

$$X \in \mathbb{R}^n$$

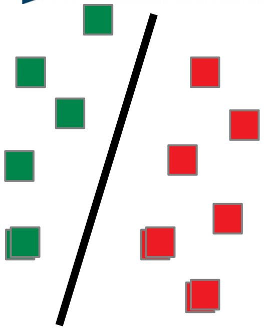
Motivation

Recall the importance of the feature extraction

In our supervised coffee cup classification:

- Image as tensor
- Feature extraction is function on tensor

Function *f* operates in feature space



(X,y)

Feature extraction

$$\begin{pmatrix} 2.06 \\ 1.76 \\ 0.06 \\ -1.7 \\ .. \\ 1.6 \end{pmatrix} \rightarrow \text{ML} \rightarrow y': \{-1,1\}$$

Feature space

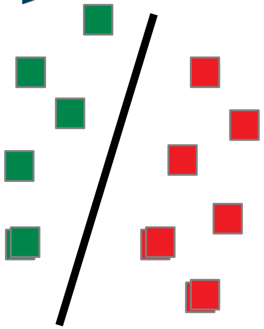
$$X \in \mathbb{R}^n$$

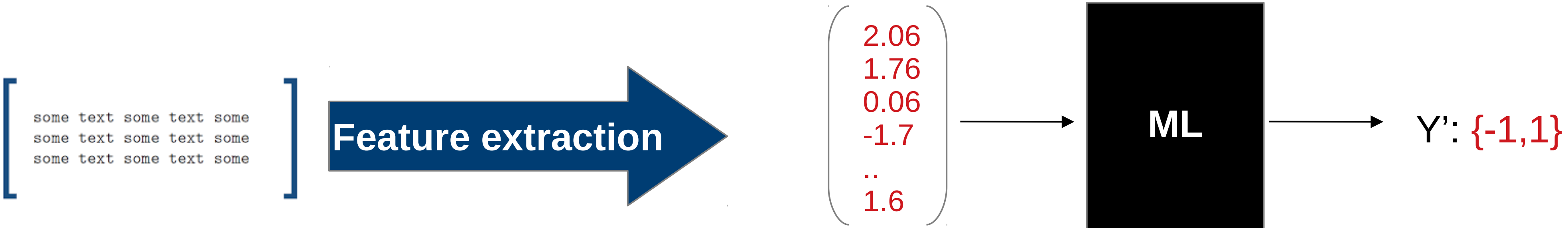
Motivation

How can we deal this non-numeric inputs?

What if we have text?

Function *f* operates in feature space





(X,y)

Feature space

$$X \in \mathbb{R}^n$$

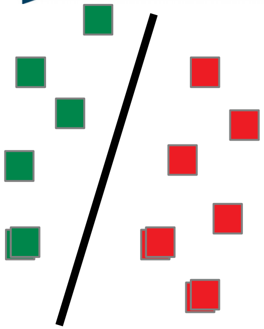
Motivation

How can we deal this non-numeric inputs?

What if we have text?

- **How can we even define a (feature) function on non-numerical inputs?**

Function *f* operates in feature space



[some text some text some
some text some text some
some text some text some]

Feature extraction

(X,y)

$\begin{pmatrix} 2.06 \\ 1.76 \\ 0.06 \\ -1.7 \\ .. \\ 1.6 \end{pmatrix}$ → **ML** → Y': {-1,1}

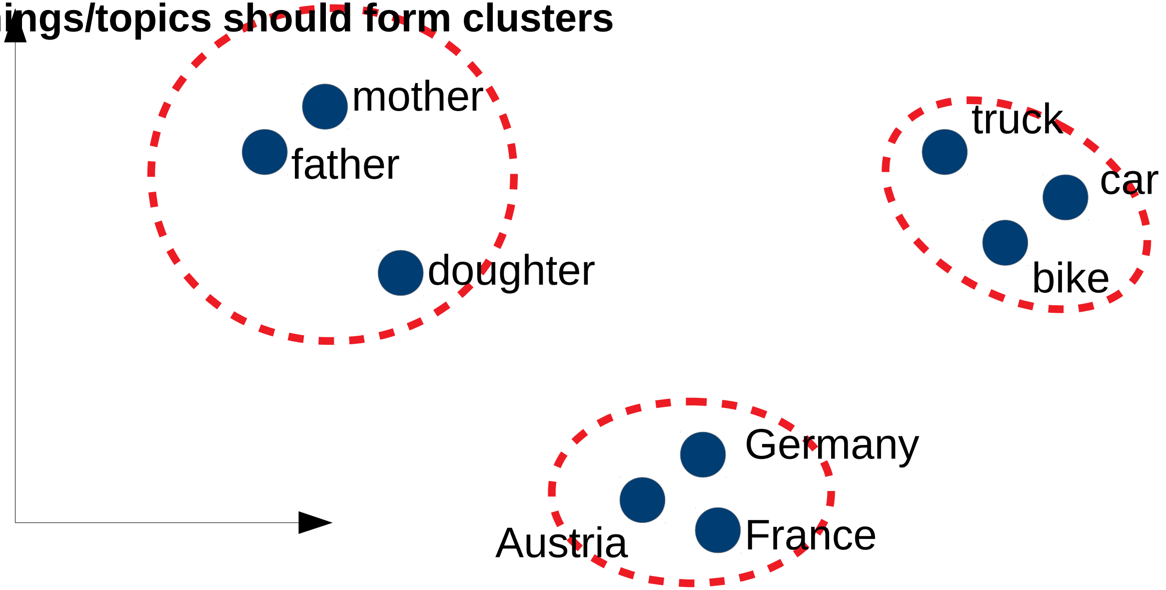
Feature space

$X \in \mathbb{R}^n$

Feature Space

Desired Properties for a “Text Space” :

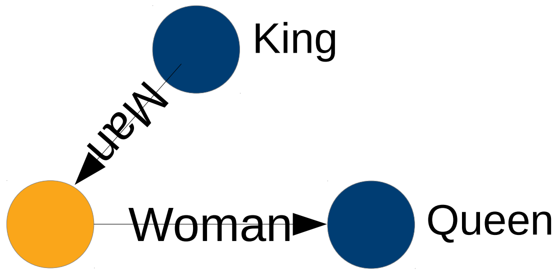
- **Unsupervised learning of feature extraction**
- **Compact, e.g. low dimension**
- **Similar words should be close**
- **Categories of things/topics should form clusters**



Feature Space

Desired Properties for a “Text Space” :

- **Unsupervised learning of feature extraction**
- **Compact, e.g. low dimension**
- **Similar words should be close**
- **Categories of things/topics should form clusters**
- **Dimensions should encode certain properties**
- **Basic “Text Calculus” would be very nice, e.g.:**

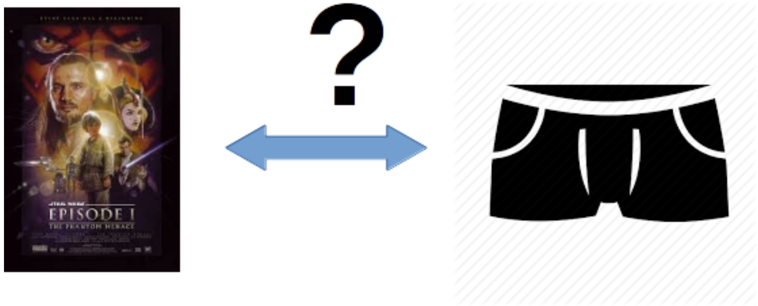


$$King - man + woman = queen$$

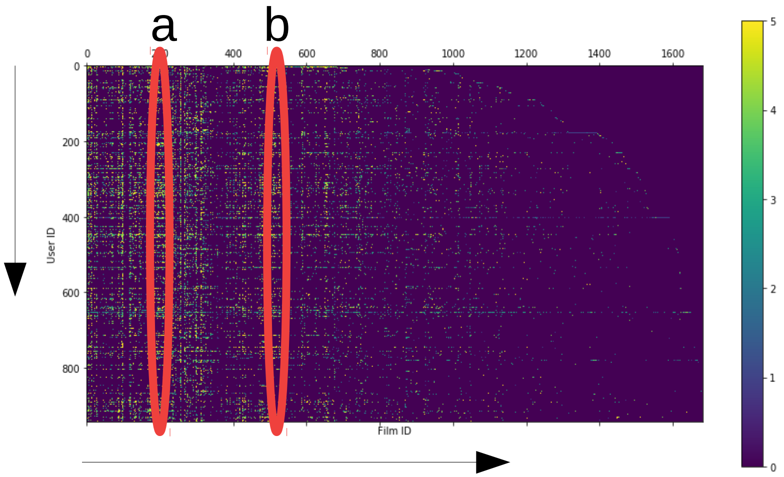
How to measure complex similarities?

Recall Collaborative Filters

Instead of using complex item-to-item measures, we used the context of very simple features, like user ratings:



vs



$$d_{cos}(\vec{a},\vec{b}) := \frac{<\vec{a},\vec{b}>}{|\vec{a}||\vec{b}|}$$

How to measure complex similarities?

Similarity by co-occurrence

Basic idea: **similar things occur in similar context**

How to measure complex similarities?

Similarity by co-occurrence

Basic idea: **similar things occur in similar context**

Text example: **which words can we fill in the blank?**

"The ________ is climbing on the tree..."

How to measure complex similarities?

Hochschule Offenburg
offenburg.university

Similarity by co-occurrence

Basic idea: **similar things occur in similar context**

Text example: **which words can we fill in the blank?**

“The _________ is climbing on the tree...“

More context:

“...His sister is 10 years old“

How to measure complex similarities?

Similarity by co-occurrence

Basic idea: **similar things occur in similar context**

For Text: **similar words have a similar context**

youngster
lad
guy
…

"The _boy_ is climbing on the tree..."

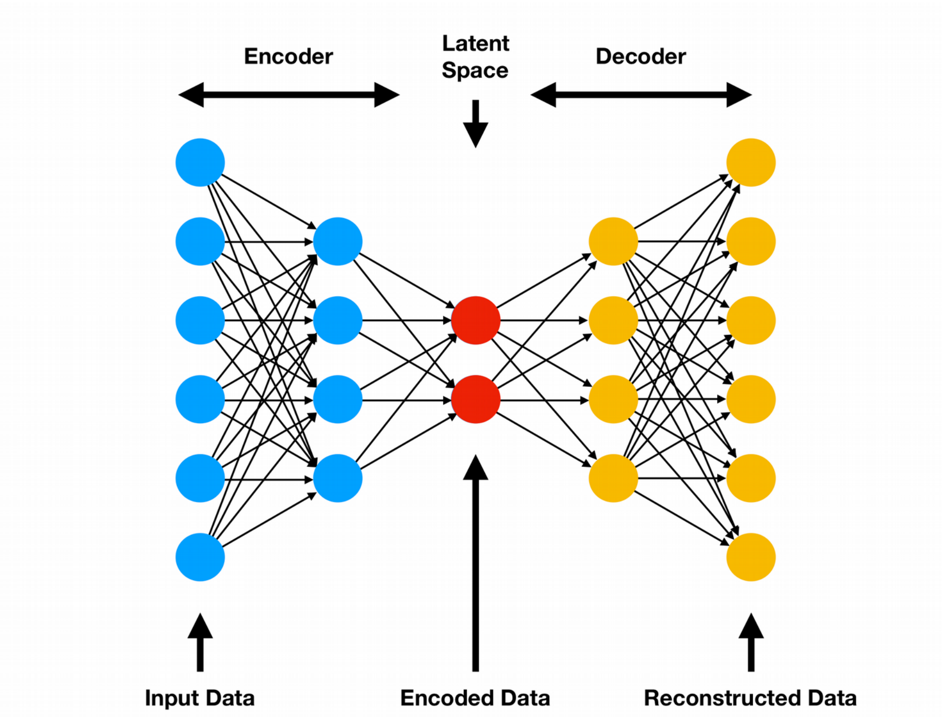
Other words that also fit are ***similar***

More context:

"...His sister is 10 years old"

Recall Auto Encoder: Latent Space

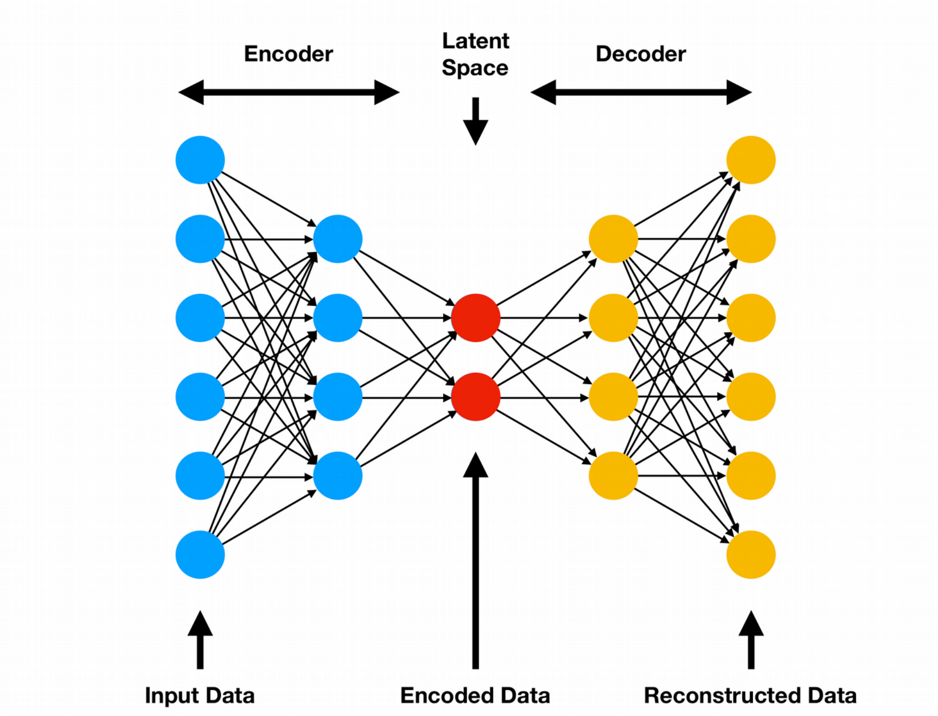
Use context to learn a space which in „embedding“ the words



- **Neural Network**
- **learning a compact latent space representation**

Recall Auto Encoder: Latent Space

Use context to learn a space which in „embedding" the words



- **Neural Network**
- **learning a compact latent space representation**
- **Main question:**

  I) What should be the optimization criteria ?

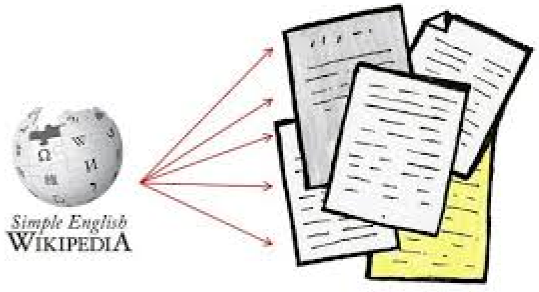
  II) How to handle Text input ?

  III) Where to get the context information ?

Q3. Where do we get the context from?

Answer: analyze a lot of text!

Open text sources:



Google News

Full Wikipedia (available for many languages)

Q2: How to present Words as numbers?

Answer: (initially) just enumerate them → compute embedding

A - Use a dictionary and count all words

B - Assign unique number to all words

C- use „one-hot-encoding“

Trivial feature space
→ has non of the desired properties!

„Junge“ → 8734 → $\begin{pmatrix}0\\0\\0\\0\\ ..\\1\\0\\ \ldots\end{pmatrix}$

The Word2Vec Algorithm

Q1: How to meet our context objective?

The famous Word2Vec Algorithm comes in two variants, *CBOW* and *skip-gram.*

The objective in in both cases, to use the context of a word to learn the embedding indirectly by constructing a prediction Task:

Efficient Estimation of Word Representations in Vector Space

Tomas Mikolov
Google Inc., Mountain View, CA
tmikolov@google.com

Kai Chen
Google Inc., Mountain View, CA
kaichen@google.com

Greg Corrado
Google Inc., Mountain View, CA
gcorrado@google.com

Jeffrey Dean
Google Inc., Mountain View, CA
jeff@google.com

Abstract

We propose two novel model architectures for computing continuous vector representations of words from very large data sets. The quality of these representations is measured in a word similarity task, and the results are compared to the previously best performing techniques based on different types of neural networks. We observe large improvements in accuracy at much lower computational cost, i.e. it takes less than a day to learn high quality word vectors from a 1.6 billion words data set. Furthermore, we show that these vectors provide state-of-the-art performance on our test set for measuring syntactic and semantic word similarities.

The Word2Vec Algorithm

Q1: How to meet our context objective?

The famous Word2Vec Algorithm comes in two variants,

CBOW **and** ***skip-gram.***

The objective in in both cases, to use the context of a word

to learn the embedding indirectly by constructing a prediction

Task: Continuous bag of words (CBOW)

→ predict a word from it's contex

Boy

"The ________ is climbing on the tree..."

The Word2Vec Algorithm

Q1: How to meet our context objective?

The famous Word2Vec Algorithm comes in two variants,
CBOW **and** ***skip-gram.***

The objective in in both cases, to use the context of a word to learn the embedding indirectly by constructing a prediction Task: Skip-gram

→ predict the contex from a word

Boy

“The ________ is climbing on the tree...“

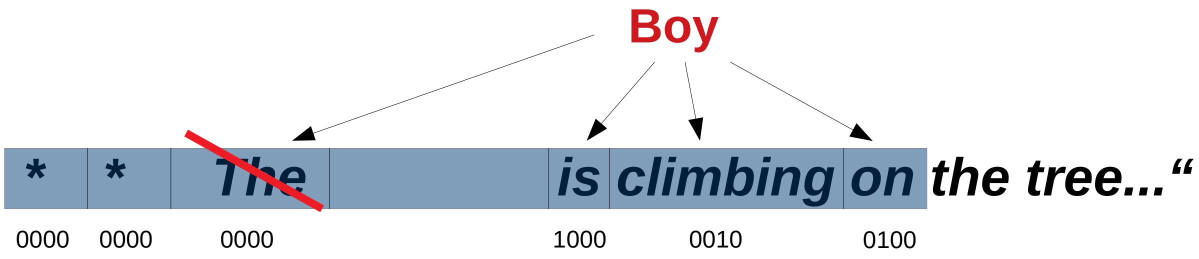
The Word2Vec Algorithm

Hochschule Offenburg
offenburg.university

Context modeling:

In Both cases:

- **Words are represented by „one-hot“ encodings of their dictionary numbers**
  - Pre-processing: remove fill-words like „a“,“the“,...
- **The context is defined by a window of a fixed size**
  - Fill window at beginning and end of text with some placeholder

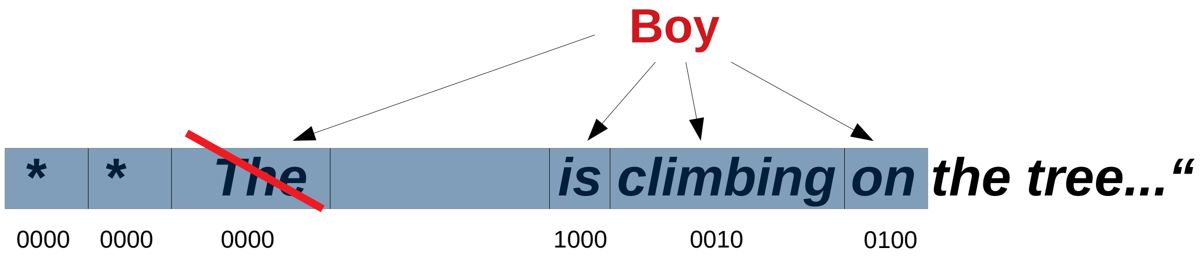


The Word2Vec Algorithm

Context modeling:

In Both cases:

- **Words are represented by „one-hot“ encodings of their dictionary numbers**
  - Pre-processing: remove fill-words like „a“,“the“,...
- **The context is defined by a window of a fixed size**
  - Fill window at beginning and end of text with some placeholder
- **Use a simple NN classification model for the prediction (next slide)**



Word2Vec

Hochschule Offenburg
offenburg.university

Neural Network Model for the *CBOW* case

*	*	~~The~~		*is*	*climbing*	*on*	*the tree...“*
0000	0000			1000	0010	0100	

Input window

As one-hot matrix

$$\begin{pmatrix} 0000 \\ 0000 \\ 0000 \\ 1000 \\ 0010 \\ 0100 \end{pmatrix}$$

Window size x number of words in vocabulary
→ Up to 100k dimensions

Word2Vec

Neural Network Model for the *Skip-Gram* case

*** * ~~The~~ is climbing on the tree…“**

0000 0000 1000 0010 0100

Input window

As one-hot matrix

One or more fully connected layers
Output: low dimension **embedding pace** → 100 – 300 dims

Word2Vec

Neural Network Model for the *Skip-Gram* case

*	*	~~The~~		is	climbing	on	the tree...“
0000	0000			1000	0010	0100	

Input window

As one-hot matrix

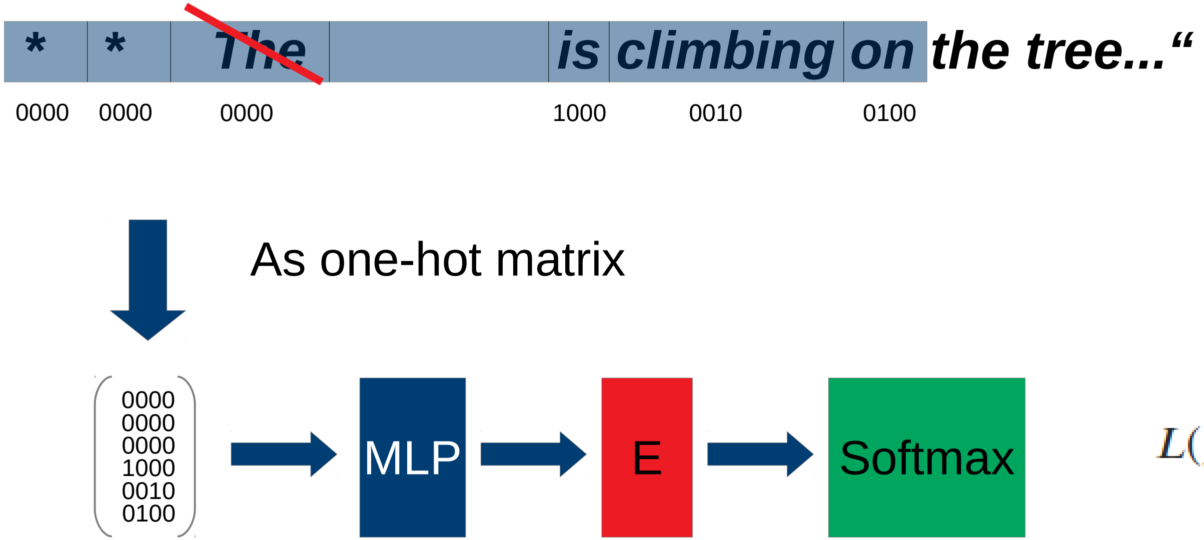
$$\begin{pmatrix} 0000 \\ 0000 \\ 0000 \\ 1000 \\ 0010 \\ 0100 \end{pmatrix} \rightarrow \text{MLP} \rightarrow \text{E} \rightarrow \text{Softmax}$$

$$L(y^i, {y^i}') := \frac{e^{{y^i}'}}{\sum_j^k e^{{y_j}'}}$$

Word2Vec

Hochschule Offenburg
offenburg.university

Neural Network Model for the *Skip-Gram* case

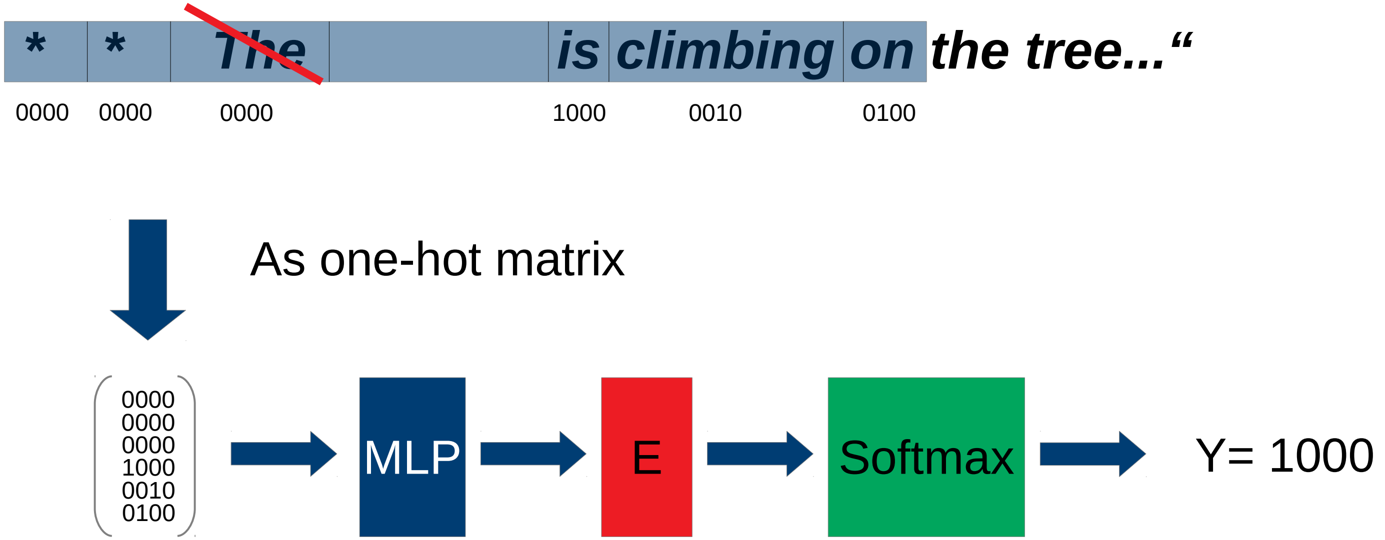


Input window

$$L(y^i, {y^i}') := \frac{e^{{y^i}'}}{\sum_j^k e^{{y_j}'}}$$

Word2Vec

Neural Network Model for the *Skip-Gram* case



Input window

Standard supervised NN training

Word2Vec

Neural Network Model for the *Skip-Gram* case

*	*	~~The~~		is	climbing	on	the tree...“
0000	0000	0000		1000	0010	0100	

Input window

As one-hot matrix

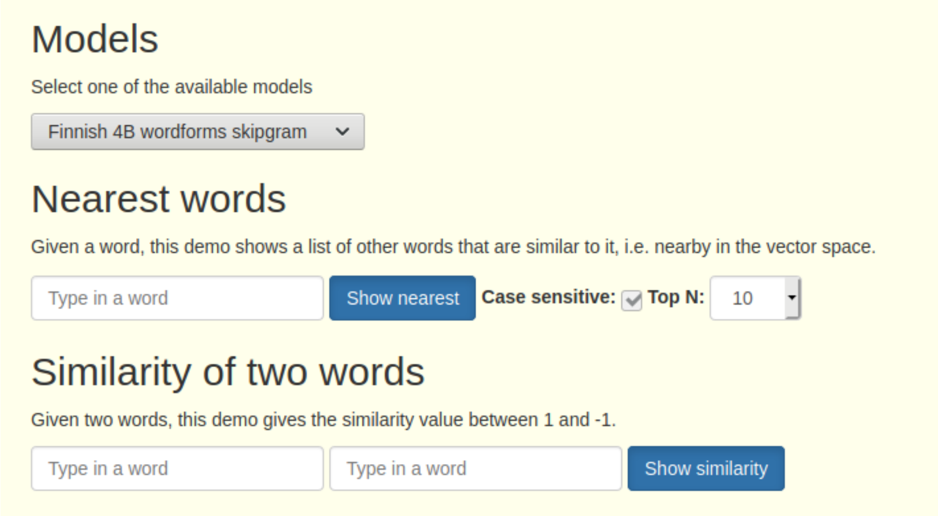
$$\begin{pmatrix} 0000 \\ 0000 \\ 0000 \\ 1000 \\ 0010 \\ 0100 \end{pmatrix} \rightarrow \text{MLP} \rightarrow \text{E} \rightarrow \text{Softmax}$$

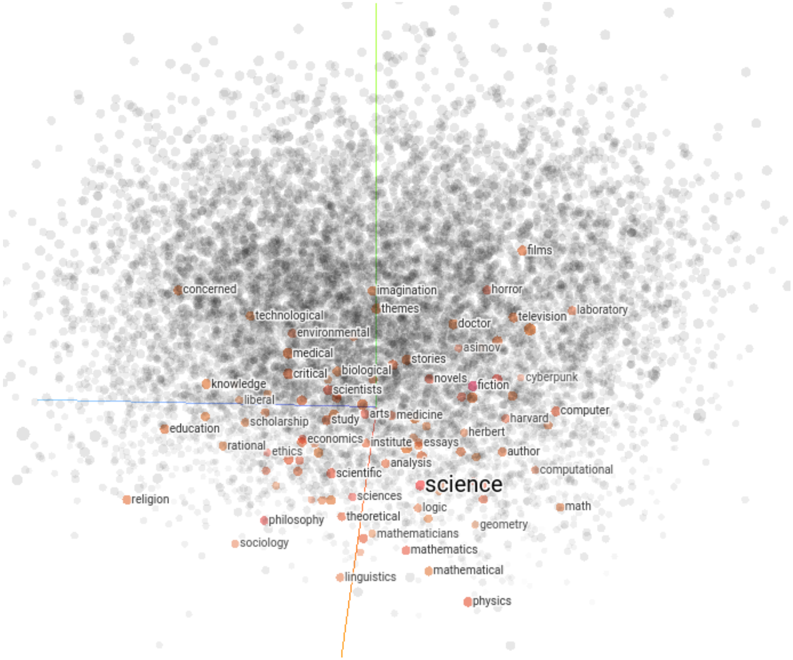
$$L(y^i, y^{i\prime}) := \frac{e^{y^{i\prime}}}{\sum_j^k e^{y_j^{i\prime}}}$$

Problem: **number** of classes is very large!

→ **solution: use random sub-set**

Word2Vec Demos





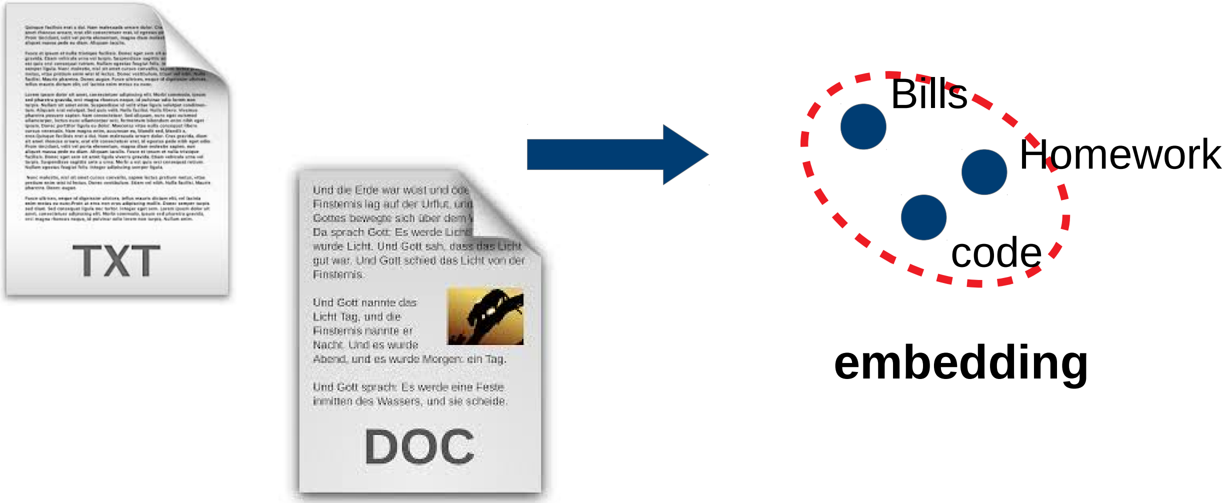
http://bionlp-www.utu.fi/wv_demo/

https://projector.tensorflow.org/

Doc2Vec

How to embed text documents – not only words?

The Doc2Vec algorithm is a simple extension of Word2Vec:



embedding

Distributed Representations of Sentences and Documents

Quoc Le QVL@GOOGLE.COM
Tomas Mikolov TMIKOLOV@GOOGLE.COM
Google Inc, 1600 Amphitheatre Parkway, Mountain View, CA 94043

Abstract

Many machine learning algorithms require the input to be represented as a fixed-length feature vector. When it comes to texts, one of the most common fixed-length features is bag-of-words. Despite their popularity, bag-of-words features have two major weaknesses: they lose the ordering of the words and they also ignore semantics of the words. For example, "powerful," "strong" and "Paris" are equally distant. In this paper, we propose *Paragraph Vector*, an unsupervised algorithm that learns fixed-length feature representations from variable-length pieces of texts, such as sentences, paragraphs, and documents. Our algorithm represents each document by a dense vector which is trained to predict words in the document. Its construction gives our algorithm the potential to overcome the weaknesses of bag-of-words models. Empirical results show that Paragraph Vectors outperform bag-of-words models as well as other techniques for text representations. Finally, we achieve new state-of-the-art results on several text classification and sentiment analysis tasks.

1. Introduction

tages. The word order is lost, and thus different sentences can have exactly the same representation, as long as the same words are used. Even though bag-of-n-grams considers the word order in short context, it suffers from data sparsity and high dimensionality. Bag-of-words and bag-of-n-grams have very little sense about the semantics of the words or more formally the distances between the words. This means that words "powerful," "strong" and "Paris" are equally distant despite the fact that semantically, "powerful" should be closer to "strong" than "Paris."

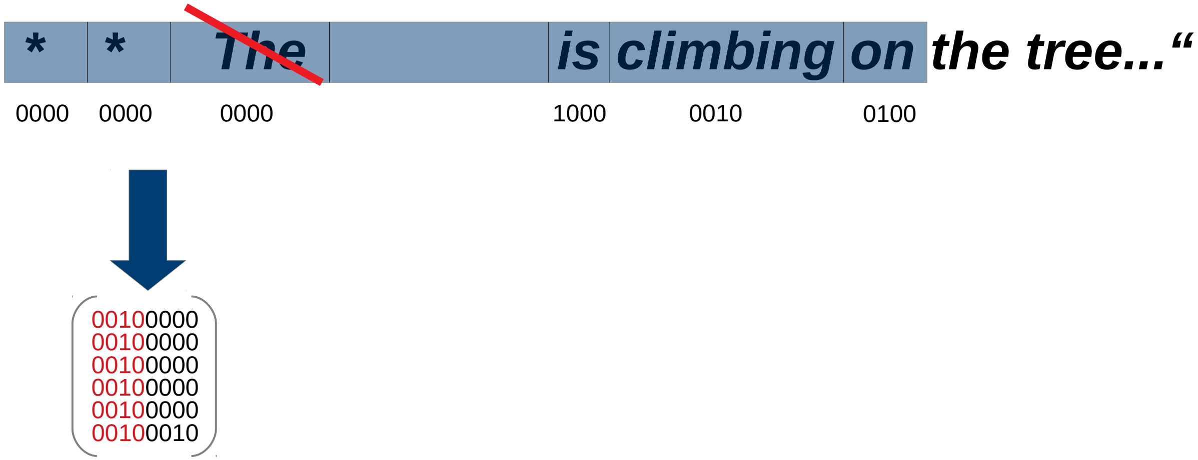
In this paper, we propose *Paragraph Vector*, an unsupervised framework that learns continuous distributed vector representations for pieces of texts. The texts can be of variable-length, ranging from sentences to documents. The name Paragraph Vector is to emphasize the fact that the method can be applied to variable-length pieces of texts, anything from a phrase or sentence to a large document.

In our model, the vector representation is trained to be useful for predicting words in a paragraph. More precisely, we concatenate the paragraph vector with several word vectors from a paragraph and predict the following word in the given context. Both word vectors and paragraph vectors are trained by the stochastic gradient descent and backpropagation (Rumelhart et al., 1986). While paragraph vectors are unique among paragraphs, the word vectors are shared. At prediction time, the paragraph vectors are inferred by fix-

Doc2Vec

How to embed text documents – not only words?

The Doc2Vec algorithm is a simple extension of Word2Vec:



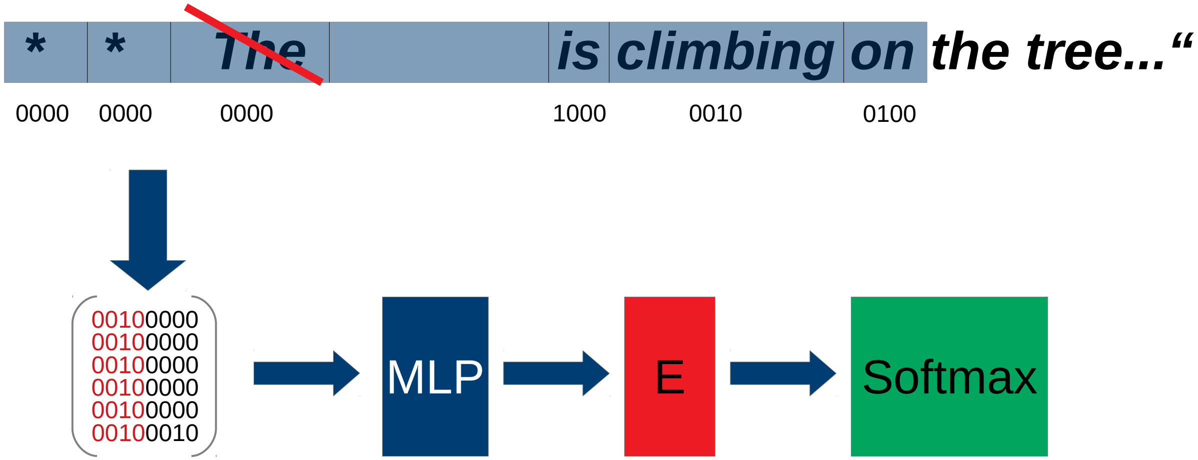
Input window

Add document ID

Doc2Vec

How to embed text documents – not only words?

The Doc2Vec algorithm is a simple extension of Word2Vec:



Input window

The rest stays the same!

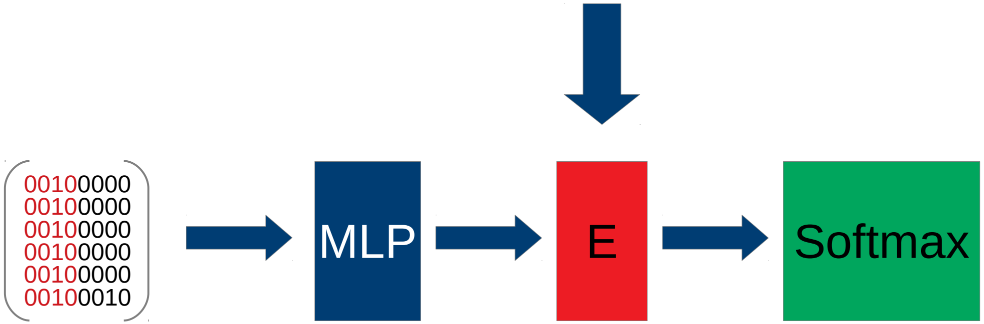
Add document ID

Doc2Vec

How to embed text documents – not only words?

Step 2: Clustering by document ID

Compute center of samples per document in embedding space

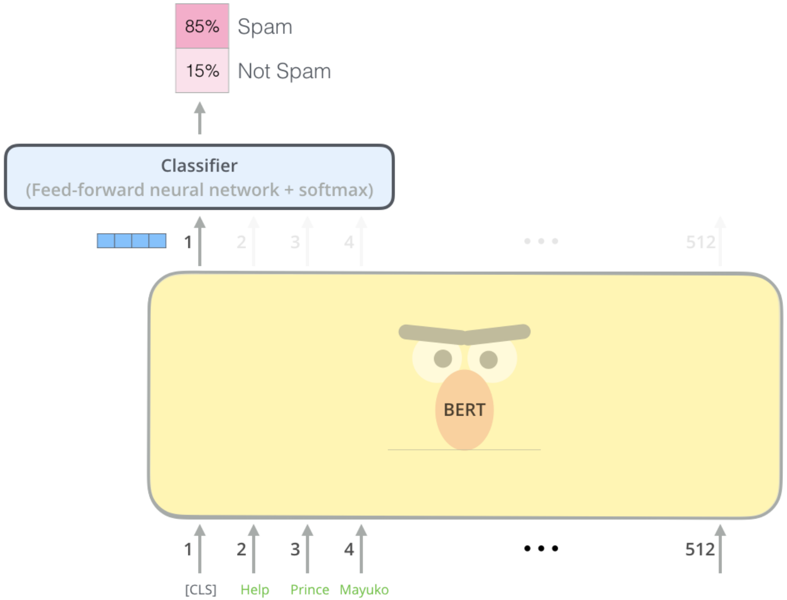


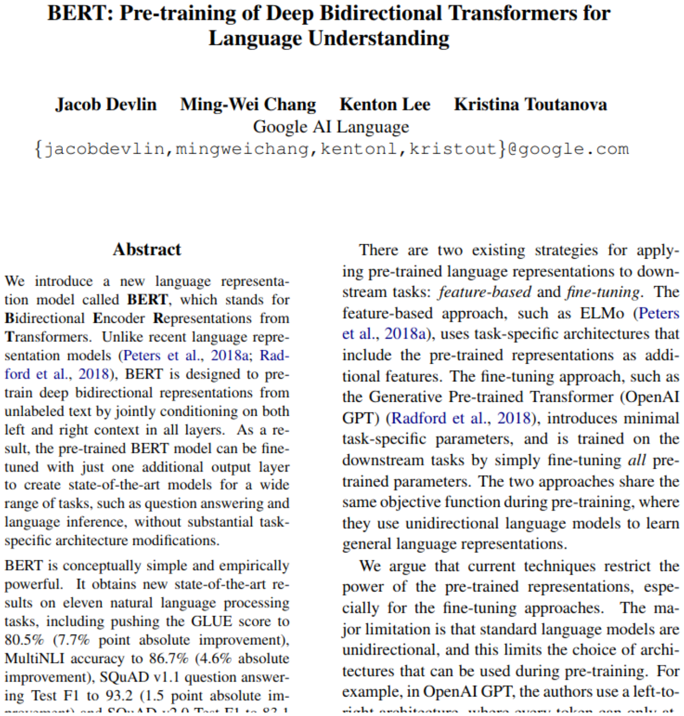
Add document ID

BERT

State of the Art Deep Learning Approach

BERT: Bidirectional Encoder Representations from Transformers





BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Jacob Devlin Ming-Wei Chang Kenton Lee Kristina Toutanova
Google AI Language
{jacobdevlin,mingweichang,kentonl,kristout}@google.com

Abstract

We introduce a new language representation model called **BERT**, which stands for **B**idirectional **E**ncoder **R**epresentations from **T**ransformers. Unlike recent language representation models (Peters et al., 2018a; Radford et al., 2018), BERT is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task-specific architecture modifications.

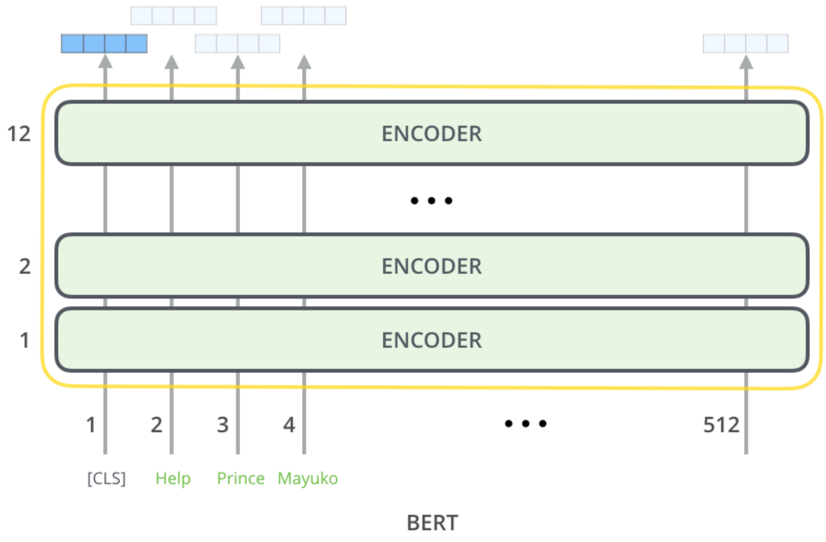
BERT is conceptually simple and empirically powerful. It obtains new state-of-the-art results on eleven natural language processing tasks, including pushing the GLUE score to 80.5% (7.7% point absolute improvement), MultiNLI accuracy to 86.7% (4.6% absolute improvement), SQuAD v1.1 question answering Test F1 to 93.2 (1.5 point absolute im-

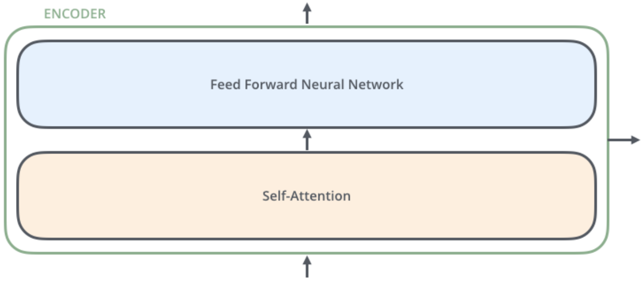
There are two existing strategies for applying pre-trained language representations to downstream tasks: *feature-based* and *fine-tuning*. The feature-based approach, such as ELMo (Peters et al., 2018a), uses task-specific architectures that include the pre-trained representations as additional features. The fine-tuning approach, such as the Generative Pre-trained Transformer (OpenAI GPT) (Radford et al., 2018), introduces minimal task-specific parameters, and is trained on the downstream tasks by simply fine-tuning *all* pre-trained parameters. The two approaches share the same objective function during pre-training, where they use unidirectional language models to learn general language representations.

We argue that current techniques restrict the power of the pre-trained representations, especially for the fine-tuning approaches. The major limitation is that standard language models are unidirectional, and this limits the choice of architectures that can be used during pre-training. For example, in OpenAI GPT, the authors use a left-to-

Figures by: http://jalammar.github.io/illustrated-bert/

Basic Architecture



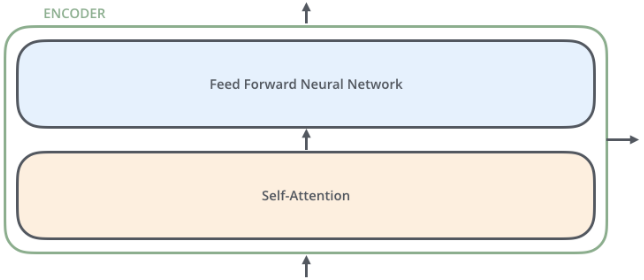


Figures by: http://jalammar.github.io/illustrated-bert/

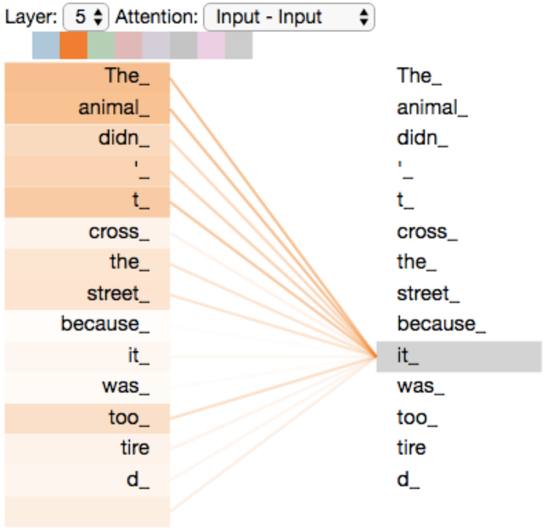
BERT

Hochschule Offenburg
offenburg.university

Attention Layers



Encoding: to which word does „*it*“ refer?

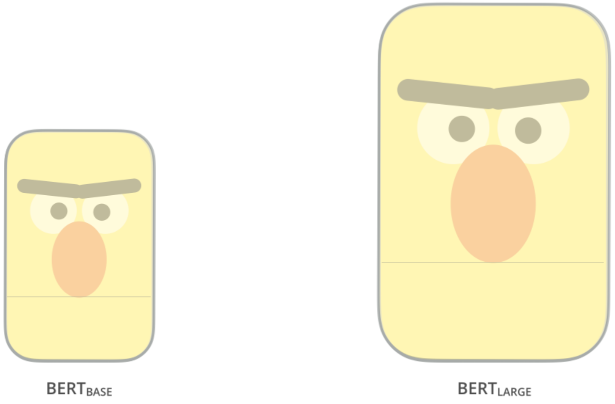


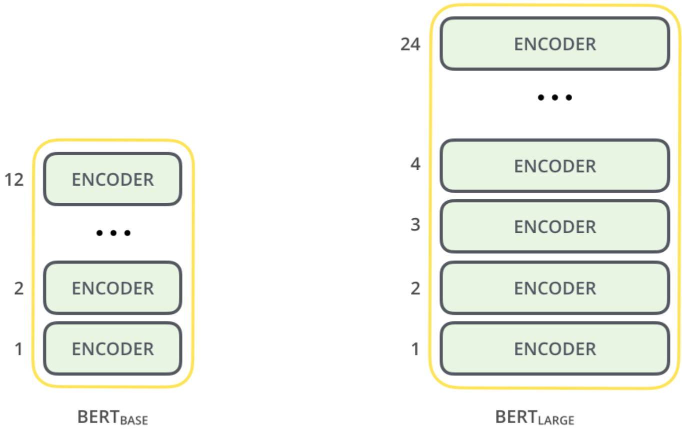
Details: https://jalammar.github.io/illustrated-transformer/

BERT

State of the Art Deep Learning Approach

Then is more than one BERT:





Figures by: http://jalammar.github.io/illustrated-bert/

BERT Resources

Hochschule Offenburg
offenburg.university

For the project work

BERT on Keras: https://github.com/CyberZHG/keras-bert

A Pre-trained Model for German BERT: https://deepset.ai/german-bert