

Time Series and NLP



Today

Time series and NLP

Core components

- Embeddings

- Recurrent Neural Nets (RNNs)



The real world is mostly time series

Predict: energy demand, inventory, network utilization

Control a robot

Identify who is speaking in a video

Translate English to Chinese, **speech to text**

Answer a question

Hold a **conversation**



Time series require special handling

Everything we have seen so far has a fixed input dimension. Or here...

Time series do not, so they are often

Truncated and padded to give a uniform size

Embedded: mapped to a vector

Modeled using recurrence



Embedding

A mapping from anything to a vector

Image2vec

Word2vec

Person2vec

Such that 'similar' items are close in the embedding space



Recurrent neural nets take advantage of invariances in the world

CNNs: nearby pixels are correlated and images are translational invariant
the kernels serve as 2-D, local, translationally invariant feature detectors

RNNs: time is 1-D and translationally invariant (“stationary”)
So predict the future based on a “hidden state” that summarizes the past

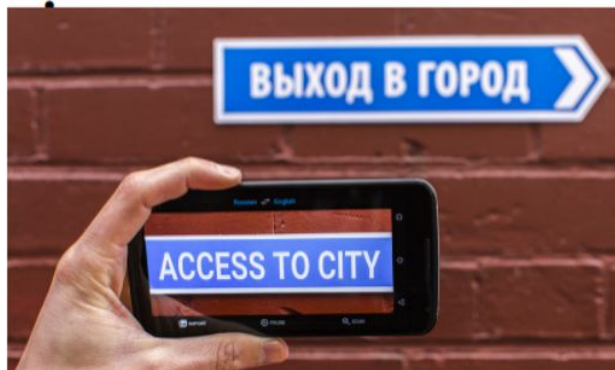


What is Natural Language Processing (NLP)?



Why Natural Language Processing?

To



To let machines communicate with us.

Image credits: google, amazon

NLP tasks

Information retrieval	query (+corpus)	→ document
Information extraction	query (+corpus)	→ fact (tuple)
Machine translation	source text	→ translation
Speech recognition	sounds	→ words
Question answering	question	→ answer
Summarization	text	→ summary
Conversational agents	prompt	→ response (and repeat)



Information Retrieval

The screenshot shows a Google search interface. The search bar contains the text "Do I need to learn deep learning?". Below the search bar, there are navigation links: "All", "News", "Images", "Videos", "Shopping", "More", "Settings", and "Tools". The search results show "About 1,240,000,000 results (1.23 seconds)". A featured snippet is displayed, stating: "You are not **required** to know **Machine Learning** for **learning Deep Learning** but I strongly recommend to first start with **Machine Learning** and then **deep** dive into **Machine Learning**. ... So in case you **do** not have that complex problem or computing resources then I **will** recommend to use **Machine Learning** algorithms first." Below this, the source is listed as "www.quora.com > Should-I-learn-machine-learning-or-deep-learning" and the title is "Should I learn machine learning or deep learning? - Quora". There are links for "About Featured Snippets" and "Feedback".

Do I need to learn deep learning?

Google

All News Images Videos Shopping More Settings Tools

About 1,240,000,000 results (1.23 seconds)

You are not **required** to know **Machine Learning** for **learning Deep Learning** but I strongly recommend to first start with **Machine Learning** and then **deep** dive into **Machine Learning**. ... So in case you **do** not have that complex problem or computing resources then I **will** recommend to use **Machine Learning** algorithms first.

www.quora.com > Should-I-learn-machine-learning-or-deep-learning

Should I learn machine learning or deep learning? - Quora

About Featured Snippets Feedback

People also ask

- Can you learn deep learning without machine learning?
- How do I start learning deep learning?
- Should I learn machine learning or deep learning first?
- Is deep learning easy to learn?

Feedback

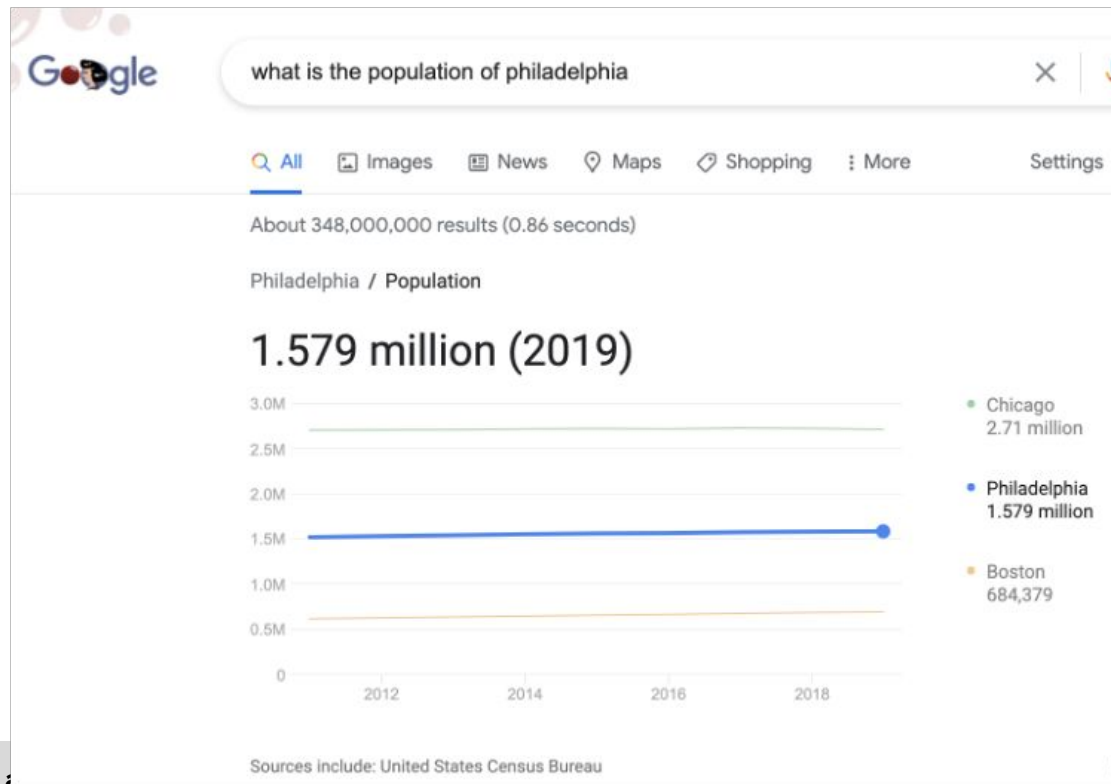
towardsdatascience.com > dont-learn-deep-learning-d23485e4c1c4

Don't Learn Deep Learning - Towards Data Science

Jun 23, 2019 - **Deep Learning** is one of the biggest breakthroughs in **machine learning** in the ... However, what **needs** to be realised is that **Deep Learning** will, ...



Information Extraction



Natural Language Generation

Input: write a message to convince someone to get a covid vaccine.

Output: Covid vaccines are essential in order to protect yourself and others from the virus. They are safe and effective, and have been shown to reduce the severity of symptoms. Get the vaccine to protect yourself and your loved ones!

Lyle Ungar
using GPT-3



NLP component tasks

Tokenizing

Part of Speech Tagging (POS tagging)

Named Entity Recognition (NER)

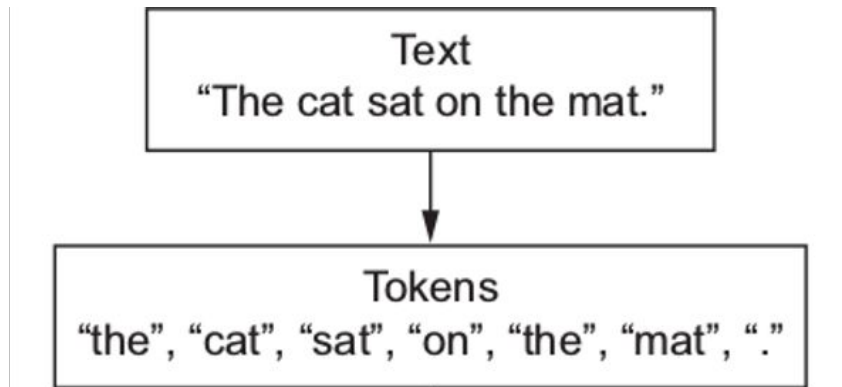
Co-reference detection

Parsing



Tokenization

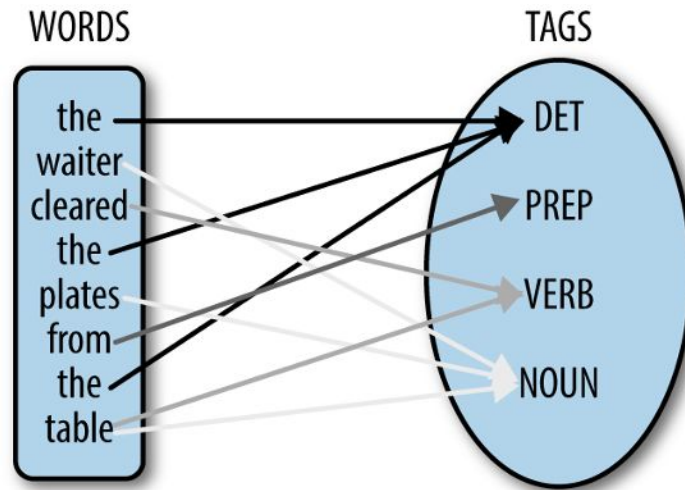
Split sequence of characters into “tokens”.



Nomenclature: the *word* “the” occurs twice above;
It is the 1st and the 5th *token*.

POS (Part-of-speech) Tagging: check copyright

POS tagging is the process of marking the words in the corpus to its corresponding part of a speech tag (noun, verb, adjective, etc), based on its context and definition.



<https://spacy.io/usage/linguistic-features>

<https://spacy.io/usage/linguistic-features>

Named entity recognition

contentSkip to site indexPoliticsSubscribeLog InSubscribeLog InToday's PaperAdvertisementSupported **ORG** byF.B.I. Agent **Peter Strzok PERSON** ,
Who Criticized Trump **PERSON** in Texts, Is FiredImagePeter Strzok, a top **F.B.I. GPE** counterintelligence agent who was taken off the special counsel
investigation after his disparaging texts about President **Trump PERSON** were uncovered, was fired. CreditT.J. Kirkpatrick **PERSON** for **The New York TimesBy Adam Goldman ORG** and **Michael S. SchmidtAug PERSON** . **13 CARDINAL** , **2018WASHINGTON CARDINAL** — **Peter Strzok PERSON** , the **F.B.I. GPE** senior counterintelligence agent who disparaged President **Trump PERSON** in inflammatory text messages and helped
oversee the **Hillary Clinton PERSON** email and **Russia GPE** investigations, has been fired for violating bureau policies, Mr. **Strzok PERSON** 's lawyer
said **Monday DATE** .Mr. Trump and his allies seized on the texts — exchanged during the **2016 DATE** campaign with a former **F.B.I. GPE** lawyer,
Lisa Page — in PERSON assailing the **Russia GPE** investigation as an illegitimate "witch hunt." Mr. **Strzok PERSON** , who rose over **20 years DATE** at the **F.B.I. GPE** to become one of its most experienced counterintelligence agents, was a key figure in **the early months DATE** of the
inquiry.Along with writing the texts, Mr. **Strzok PERSON** was accused of sending a highly sensitive search warrant to his personal email account.The
F.B.I. GPE had been under immense political pressure by Mr. **Trump PERSON** to dismiss Mr. **Strzok PERSON** , who was removed **last summer DATE** from the staff of the special counsel, **Robert S. Mueller III PERSON** . The president has repeatedly denounced Mr. **Strzok PERSON** in posts on

<https://arxiv.org/abs/1511.08308>



Coreference

Tom was happy that **he** got a present.

Tom gave **Bill** a present but **he** didn't like it.



Language models

Language Models compute the probability of the next word.

More formally,

Given a sequence of words $x^{(1)} \dots x_{(t)}$, compute the probability of the occurrence of $x_{(t+1)}$

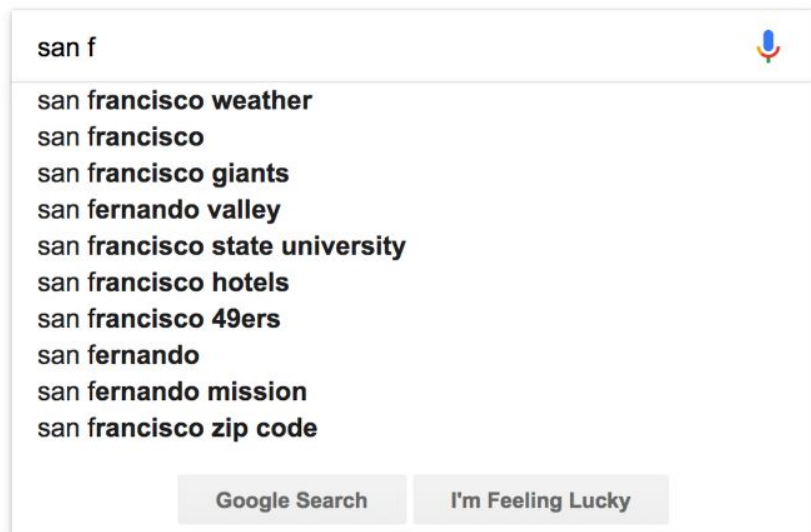
$$P(x^{(t+1)} | x^{(t)}, \dots, x^{(1)})$$

where $x^{(t+1)}$ can be any word in the vocabulary $V = \{w_1, \dots, w_{|V|}\}$

<https://www.cs.bgu.ac.il/~elhadad/nlp18/nlp02.html>



Language model use: autocompletion

A screenshot of a Google search interface. The search bar contains the text "san f". Below the search bar, a list of suggestions is displayed. At the bottom of the suggestions box are two buttons: "Google Search" and "I'm Feeling Lucky".

san f

- san francisco weather
- san francisco
- san francisco giants
- san fernando valley
- san francisco state university
- san francisco hotels
- san francisco 49ers
- san fernando
- san fernando mission
- san francisco zip code

Google Search I'm Feeling Lucky

Probability of a sentence

Language models also assign probability to the entire text.

Given text containing words $x^{(1)} \dots x^{(T)}$. Then the probability of that text is

$$\begin{aligned} P(\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(T)}) &= P(\mathbf{x}^{(1)}) \times P(\mathbf{x}^{(2)} | \mathbf{x}^{(1)}) \times \dots \times P(\mathbf{x}^{(T)} | \mathbf{x}^{(T-1)}, \dots, \mathbf{x}^{(1)}) \\ &= \prod_{t=1}^T P(\mathbf{x}^{(t)} | \mathbf{x}^{(t-1)}, \dots, \mathbf{x}^{(1)}) \end{aligned}$$

Natural Language Processing

Details of tokenization



Representations

Words/tokens: context-free or context-sensitive embeddings

Sub-word encoding (e.g., Byte Pair Encoding, BPE): frequent character sequences are treated as “words”

“bbibtech” might be three words “b” + “bib” + “tech”

Special tokens:

sep_token: separates different strings

unk_token: unknown token

pad_token: padding

cls_token: token for the entire sequence



Typical NLP pipeline

- 1) **Tokenize (or extract byte pair encodings)**
- 2) **Map tokens to embeddings using something trained on a huge corpus (word2vec, Bert,...)**
- 3) **Train a neural net with embeddings as the input**
 - a) optionally: “fine tune” the embeddings to better fit the task



Embeddings rule!



We often map objects to vectors

Similar objects should be close in the embedding space

Many things can be embedded

- embed images by using the penultimate output of the CNN

- embed a word, sentence or document

- embed a product or a person

Word embeddings

The simplest word embeddings (word2vec, Glove), map each word to a 300 dimensional vector such that words that are tend to show up in the same contexts have embeddings that are close to each other.



Distributional similarity

Words that occur in similar contexts are similar

He ate the **sandwich**.

He ate the **shrudlu**.

The **Shrudla** ate the sandwich.

“sandwich” and “shrudlu” are distributionally similar, as are “he” and “Shrudla”



One-Hot Word Representation

Before the deep learning era, we tended to treat each word as a separate symbol.

one-hot encoding example

Hotel = [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]

Motel = [0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0]

But all one-hot words are equally similar

How can we know that two words (or sentences) mean similar things?

Vector embeddings as dimensionality reduction: LSA

Latent Semantic Analysis (LSA):

context is the document the words appear in

Run SVD on a word x document matrix

Maps every word to a k -dimensional vector so that words that tend to appear in the same documents will be close

related words will be close: doctor, hospital, nurse, cancer

LSA

- 1) I ate ham and cheese
- 2) You ate cheese and crackers
- 3) he went to hospital with covid
- 4) she came home from the hospital after her operation

	Document				
	1	2	3	4	...
ate	1	1	0	0	...
cheese	1	1	0	0	...
I	1	0	0	0	...
hospital	0	0	1	1	...



Vector embeddings as dimensionality reduction: word2vec

Modern vector embeddings (word2vec)

context is the words to the left and right of each token

Run SVD on word x context matrix

Maps every word to a k -dimensional vector so that words that tend to appear in the same contexts will be close

similar words will be close: doctor, nurse

Eigenwords (like word2vec)

I ate ham
You ate cheese
You ate

	Word Before					Word After				
	ate	cheese	ham	I	You	ate	cheese	ham	I	You
ate	0	0	0	1	2	0	1	1	0	0
cheese	1	0	0	0	0	0	0	0	0	0
ham	1	0	0	0	0	0	0	0	0	0
I	0	0	0	0	0	1	0	0	0	0
You	0	0	0	0	0	2	0	0	0	0



Similar words are close

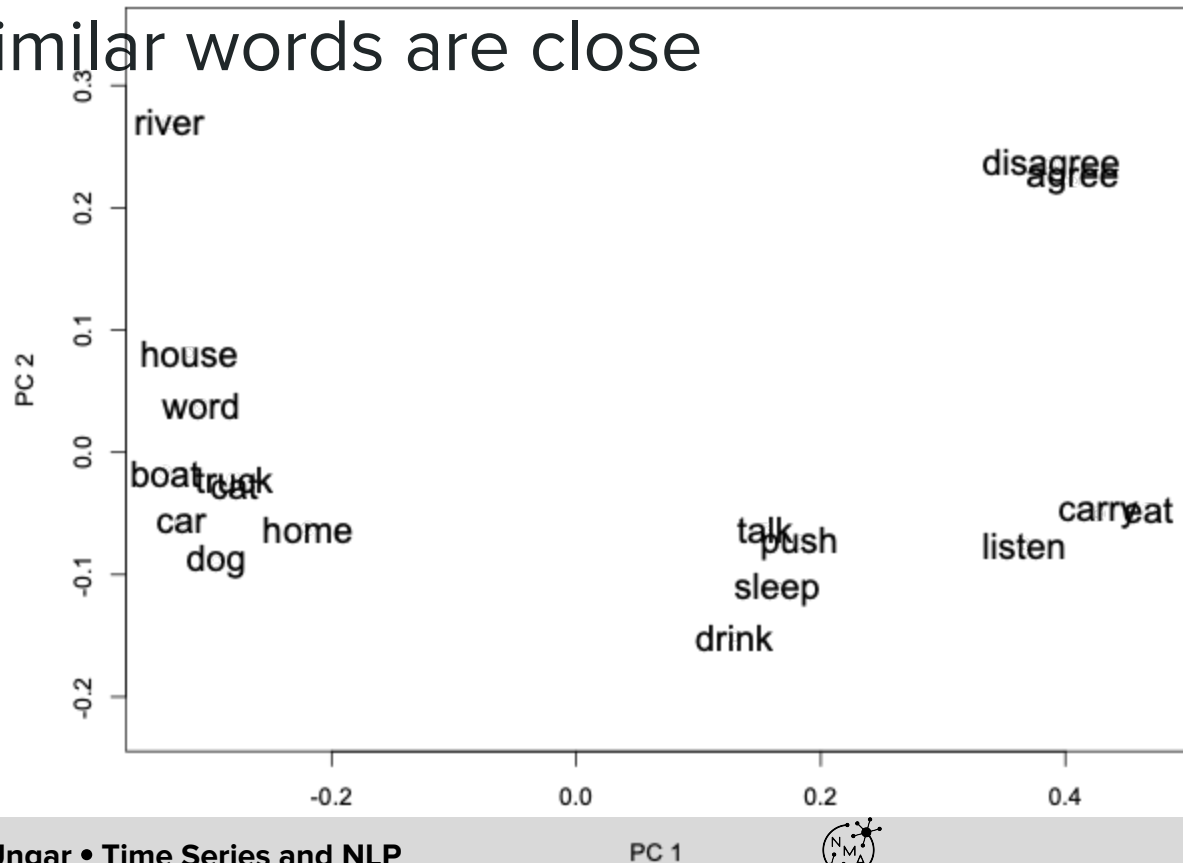


Image credit: lyle ungar



One can embed words in different languages into the same space

similar words will be close: doctor, nurse

Or even embed images and words into the same space



Distributional Similarity and Vector Embeddings

Context oblivious and context
sensitive



Context free vs. context sensitive embeddings

Words have different meanings depending on their context



Distributional similarity (again)

Words that occur in similar contexts are similar

He drank the port.

He visited the port.

He's on the port (not starboard!) side of the boat.

He needs to port the code to pytorch.



What we want of a good similarity

Similar words should be close in embedding space

But be careful: what does “similar” mean
meaning?
style?
emotion?

For now: similar means “distributionally similar”

Later: we will “fine tune” embeddings for different tasks



Two kinds of embeddings

context oblivious: embed words

LSA, word2vec, Glove, fastText

map each of e.g. 1,000,000 words to a 300-D vector

context sensitive: embed tokens in context

Bert, Elmo, and friends

map token and surrounding tokens to a 768-D vector

Many of these actually use Byte Pair Encoding



Using Embeddings



Embeddings: what to use when

context oblivious: mostly of historical interest for text

LSA, word2vec, fastText

Still used for images

context sensitive: pick a popular embedding from Hugging Face

BERT and other transformer models

These use a context oblivious Byte Pair or Word Part Encoding as input



Hugging Face



Tasks

Problems solvers

Thousands of creators work as a community to solve Audio, Vision, and Language with AI.



Image Classification

488 models



Object Detection

28 models



Question Answering

1745 models



Summarization

409 models



Text Classification

6721 models



Translation

1584 models



Tasks

- Image Classification
- Translation
- Image Segmentation
- Fill-Mask
- Automatic Speech Recognition
- Token Classification
- Sentence Similarity
- Audio Classification
- Question Answering
- Summarization
- Zero-Shot Classification

+ 16 Tasks

Libraries

- PyTorch
- TensorFlow
- JAX
- + 25

Datasets

- common_voice
- wikipedia
- squad
- glue
- bookcorpus
- c4
- emotion
- conll2003
- + 1006

Languages

- en
- es
- fr
- de
- zh
- ja
- sv
- ru
- + 178

Licenses

- apache-2.0
- mit
- cc-by-4.0
- + 36

Models 48,311

Sort: Most Download

gpt2

Text Generation • Updated May 19, 2021 • ↓ 54.9M • ♥ 109

distilgpt2

Text Generation • Updated 7 days ago • ↓ 21.4M • ♥ 54

bert-base-uncased

Fill-Mask • Updated May 18, 2021 • ↓ 17.7M • ♥ 149

distilbert-base-uncased-finetuned-sst-2-english

Text Classification • Updated Mar 22 • ↓ 13.5M • ♥ 57

roberta-base

Fill-Mask • Updated Jul 6, 2021 • ↓ 10.6M • ♥ 36

SEBIS/code_trans_t5_small_program_synthese_transfer_learning_finetune

Summarization • Updated Jun 23, 2021 • ↓ 7.14M • ♥ 2

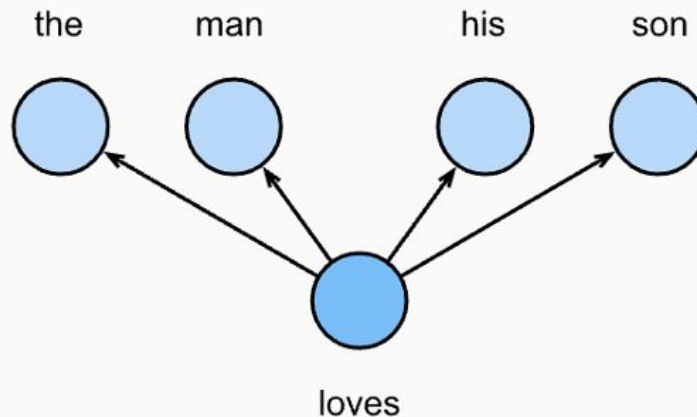
distilbert-base-uncased

Fill-Mask • Updated about 19 hours ago • ↓ 7.13M • ♥ 62

bert-base-cased

Skip-gram-based embeddings

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



Skip-gram-based embeddings

maximize the log-likelihood of neighboring words

$$-\sum_{t=1}^T \sum_{-m \leq j \leq m, j \neq 0} \log P(w^{(t+j)} | w^{(t)}).$$

where probabilities are estimated based on the embedding of the word and its context words

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

use “negative sampling” to approximate the background distribution

Continuous Bag of Words (CBOW)

Flips the dependencies

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

