Member-only story

How Negative Sampling work on word2vec?

Edward Ma · Follow

4 min read · May 12, 2019

Listen Share More

Photo by Tim Bennett on Unsplash

During neural network training, it always adjust all neuron weight so that it learn how to do the prediction correctly. In NLP, we may face more than 100 k (or even 1M) words and it will cause performance (in term of time) concern. How can we reduce the training sample in a better way ?We have hierarchical softmax previously but word2vec introduces negative sampling methodology to resolve this problem.

What is that?

When we try to predict word (we call it as context), within a certain window (e.g. 5), those word are considered as positive word. So that we can either use those positive word to predict the context word (CBOW) or using context word to predict positive word (skip-garm).

However, we need some false label to do the prediction. So we need to figure out some negative case for that. So we can pick some non-surrounding words (we call it as negative word) to be a false label.

How do we select negative samples?

“small eggs on table and in the cup” by Gaelle Marcel on Unsplash

To reduce the number of neuron weight updating to reduce training time and having a better prediction result, negative sampling is introduced in word2vec . For example, we have 10 positive words and 1 predicting words, then the total number of neuron weight updating operations is 11 instead of updating whole corpus’s neuron weight.

Will introduce two way to do the sampling which are simple sampling and adjusted sampling

For example, we have a bag including 3 apples, 10 oranges and 1 banana. If we pick 1 fruit from bag, the probability of orange is 0.71 (10/14), apple is 0.21 (3/14), banana is 0.07 (1/14).

Adjusted Sampling

Since we do not want to pick a high frequency words all the time as those words usually provide less value than rare words. In word2vec c implementation, they apply the power of 3/4 to the probability. For example, the probability of orange is 0.71 which is converted to 0.77 while banana’s probability is converted from 0.07 to 0.14. In that cases, the probability of banana is doubled.

```
# Copy from gensim

power = 0.75

for word_index in xrange(vocab_size):
    train_words_pow +=
wv.vocab[wv.index2word[word_index]].count**power
cumulative = 0.0
for word_index in xrange(vocab_size):
    cumulative += wv.vocab[wv.index2word[word_index]].count**power
    self.cum_table[word_index] = round(cumulative / train_words_pow *
domain)
```

What if the predicting word is picked from negative sampling? According to gensim implementation, it will try to pick another word again.

```
# Copy from gensim

word_indices = [predict_word.index]
while len(word_indices) < model.negative + 1:
    w =
model.cum_table.searchsorted(model.random.randint(model.cum_table[-1])
)
    if w != predict_word.index:
        word_indices.append(w)
```

Hierarchical Softmax

"low angle photography of colorful building" by Joe LIU on Unsplash

What is the benefit of using hierarchical softmax? By design, it is a binary tree, so the computational complexity reduce to log2 V (log base is 2)

→ O(V) to O(log2 V)

High frequency words are assigned short codes and it spends less time to access it high frequency words then rare words. But it is also a disadvantage as it takes longer time to those low frequency words. That

Assume that the node is word vector, it is -1 (or 0) if we go left while it is 1 if going right.

```
# Copy from gensim

sgn = (-1.0) ** predict_word.code
lprob = -log(expit(-sgn * prod_term))
model.running_training_loss += sum(lprob)
```

Code

If you are using gensim, only need to define whether using negative sampling or hierarchical softmax by passing parameter is okay.

```
# Copy from gensim API

hs (int {1,0}) - If 1, hierarchical softmax will be used for model
training. If set to 0, and negative is non-zero, negative sampling
will be used.
negative (int) - If > 0, negative sampling will be used, the int for
negative specifies how many “noise words” should be drawn (usually
between 5-20). If set to 0, no negative sampling is used.
```

General Practice

- According to paper, it suggests that using 5 ~ 20 negative words in smaller data set while only using 2–5 words for large dataset

Like to learn?

I am Data Scientist in Bay Area. Focusing on state-of-the-art in Data Science, Artificial Intelligence , especially in NLP and platform related. Feel free to connect with me on LinkedIn or following me on Medium or Github.

Machine Learning | Deep Learning | Data Science | NLP | Artificial Intelligence

Follow

Written by Edward Ma

3.3K Followers

Focus in Natural Language Processing, Data Science Platform Architecture. https://makcedward.github.io/

More from Edward Ma

Edward Ma in Towards Data Science

Data Augmentation in NLP

Introduction to Text Augmentation

5 min read · Apr 12, 2019

500 2

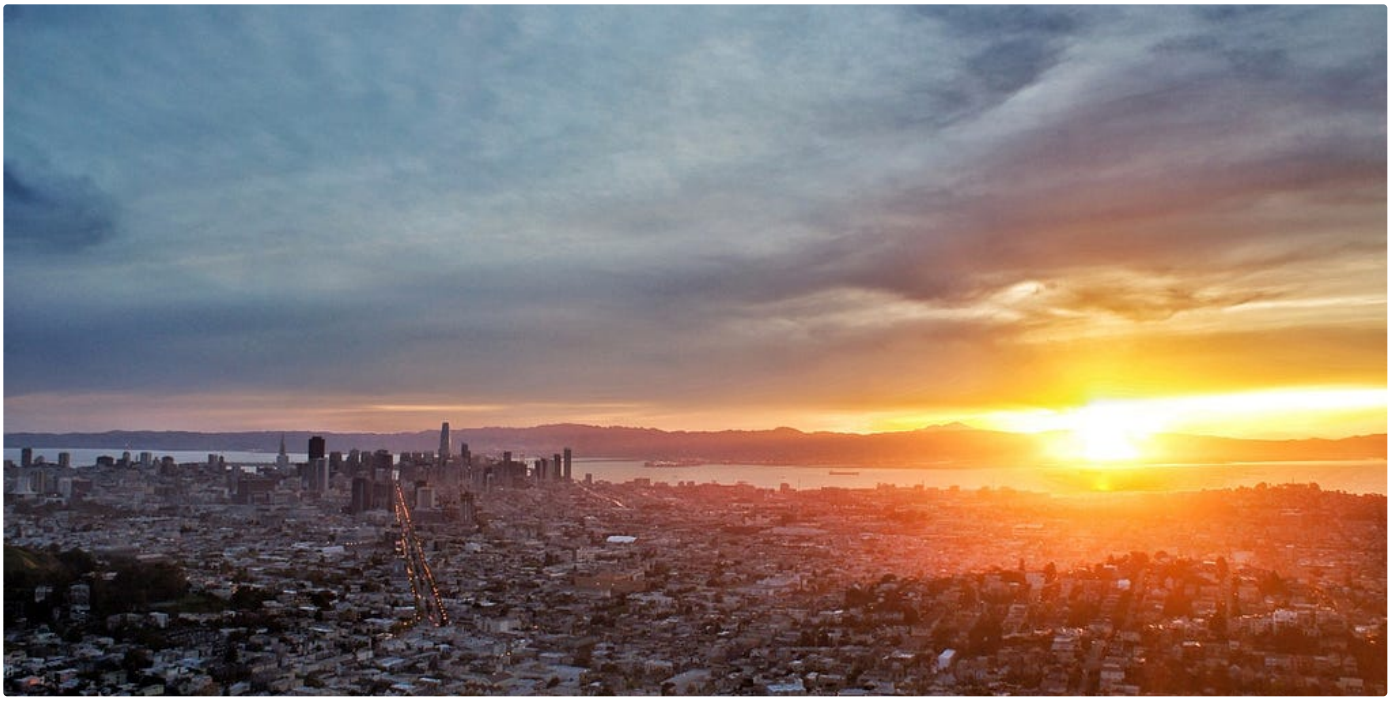
Edward Ma

Data Augmentation for Audio

Data Augmentation

3 min read · Jun 1, 2019

227 3



Edward Ma in Towards Data Science

Data Augmentation library for text

In previous story, you understand different approaches to generate more training data for your NLP task model. In this story, we will...

7 min read · Apr 20, 2019

Edward Ma in Towards Data Science

3 basic Distance Measurement in Text Mining

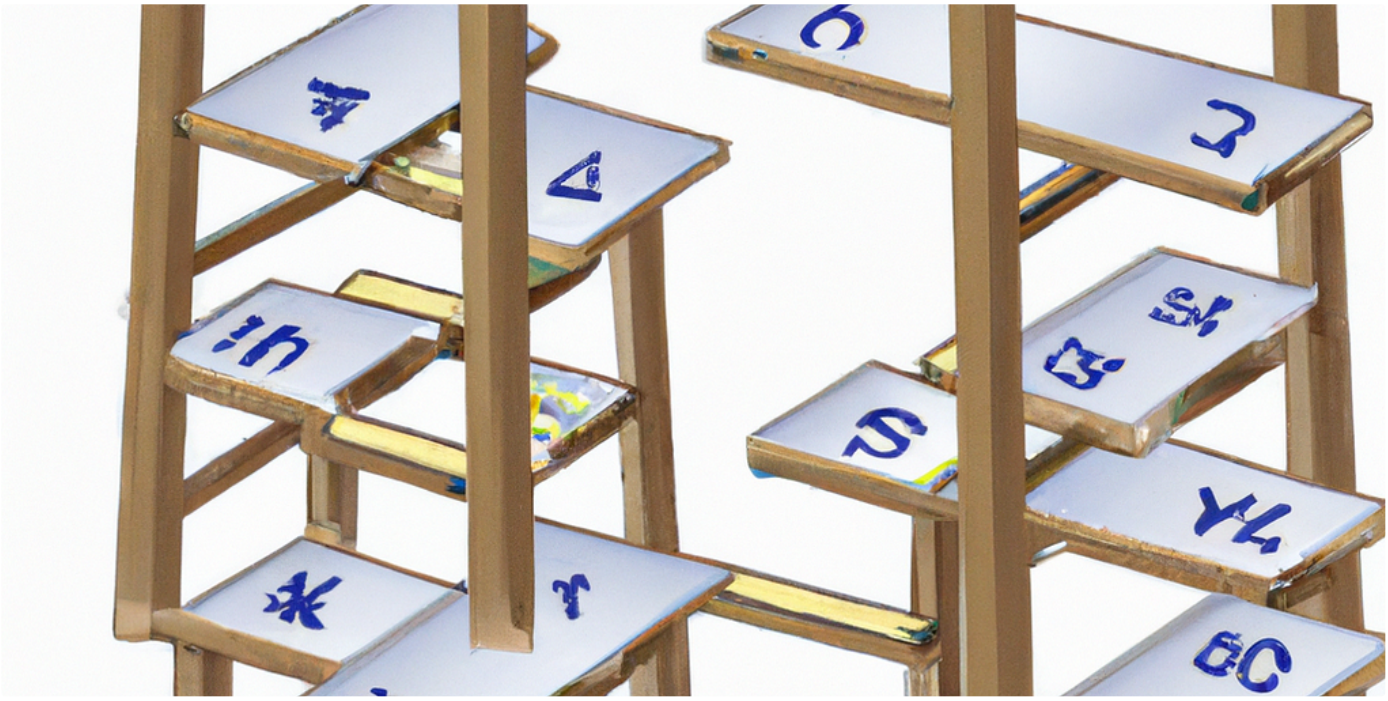
In NLP, we also want to find the similarity among sentence or document. Text is not like number and coordination that we cannot compare the...

· 4 min read · Jul 5, 2018

338 1

See all from Edward Ma

Recommended from Medium

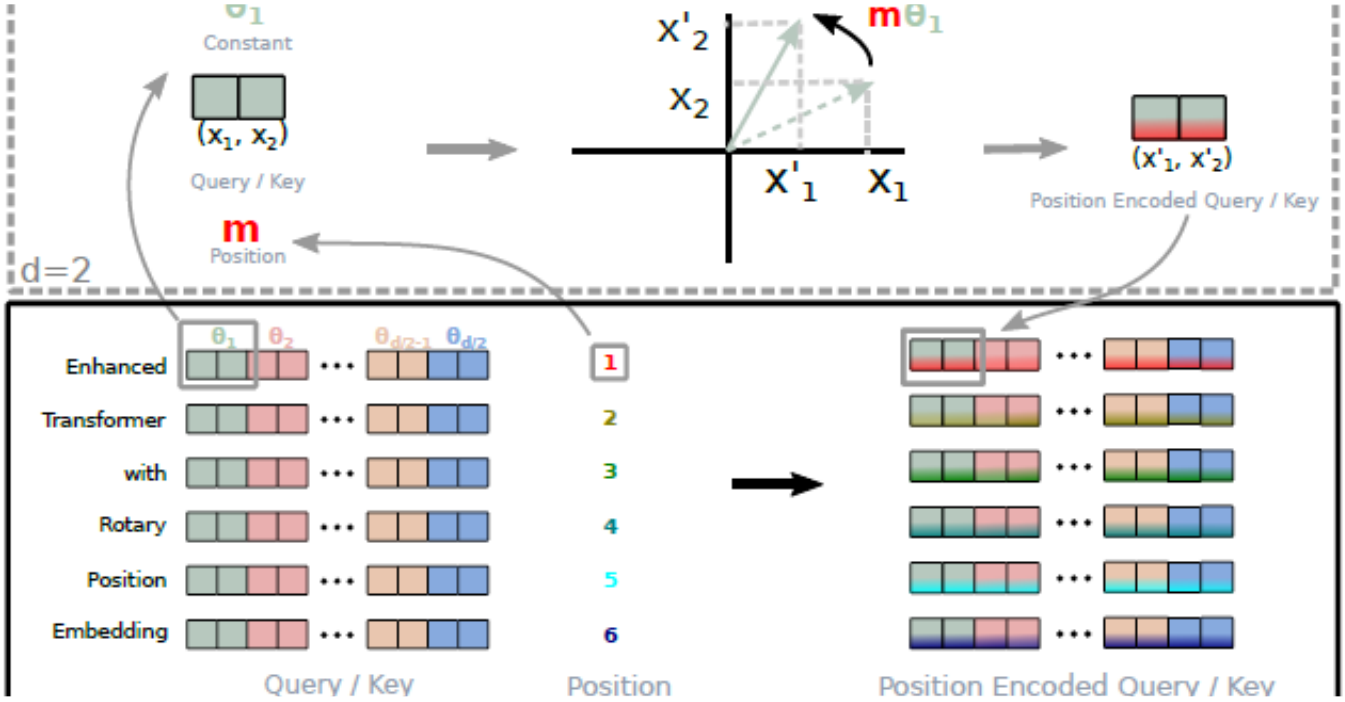


Will Badr in Towards Data Science

The Secret to Improved NLP: An In-Depth Look at the nn.Embedding Layer in PyTorch

Dissecting the `nn.Embedding` layer in PyTorch and a complete guide on how it works

· 8 min read · Jan 24



Sik-Ho Tsang

Brief Review—RoFormer: Enhanced Transformer with Rotary Position Embedding

RoFormer: Rotary Position Embedding (RoPE), for Position Information

5 min read · Dec 17, 2022

1

Lists

What is ChatGPT?

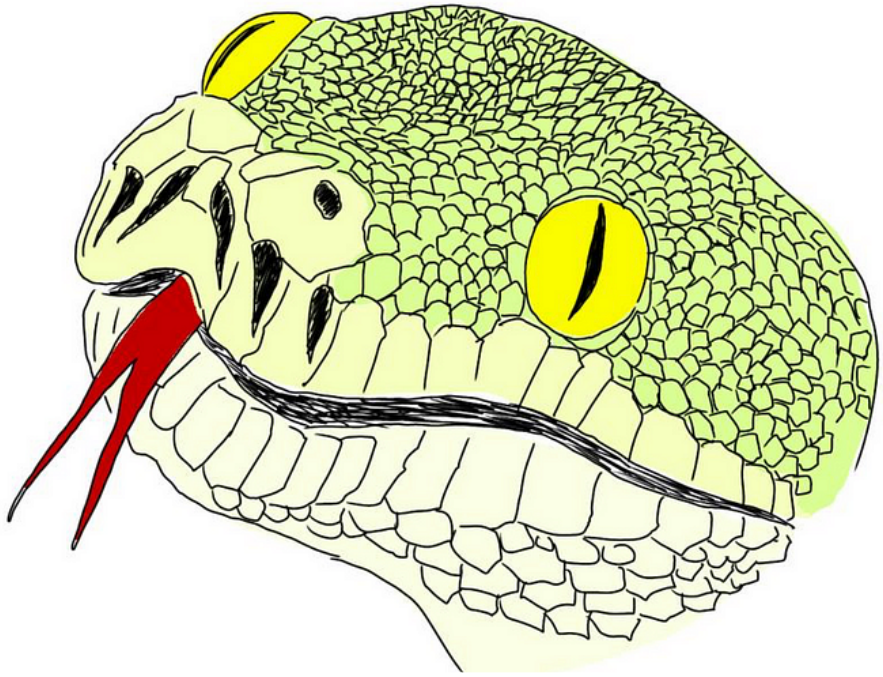
9 stories · 81 saves

Staff Picks

339 stories · 95 saves

Stories to Help You Level-Up at Work

19 stories · 74 saves

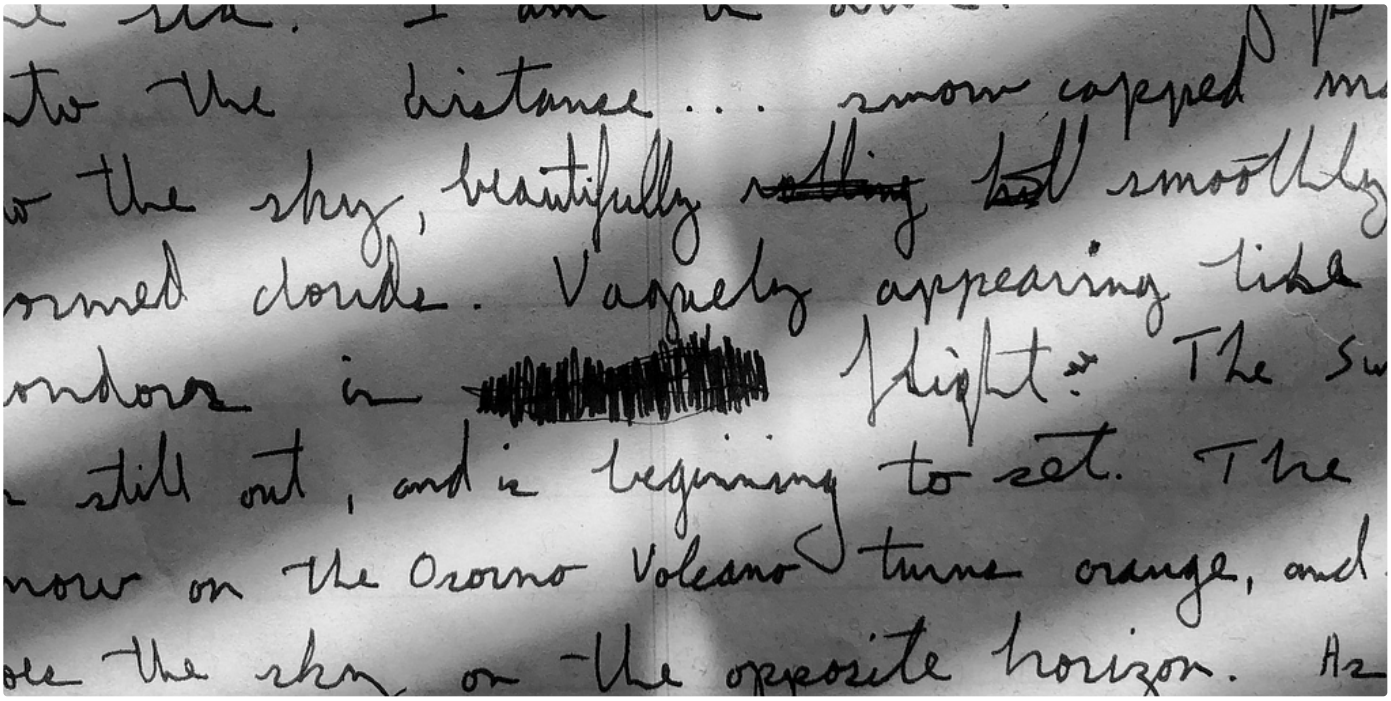


Liu Zuo Lin in Python in Plain English

Python Word2Vec For Text Classification (With LSTM)

Plug & Play Code For Those With No Time

· 5 min read · Feb 4



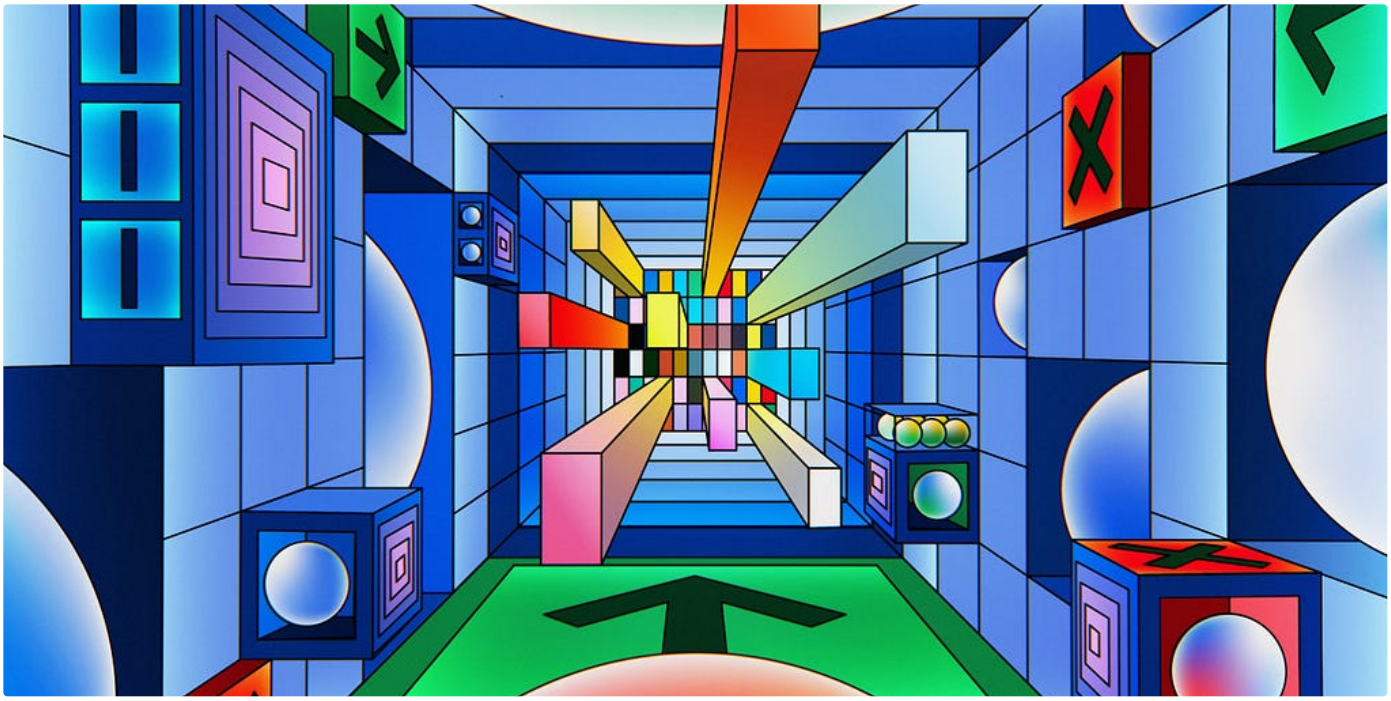
Panu Korhonen in UX Collective

Train yourself, not the model: a designer's view on language models

This post discusses our experiences in designing healthcare applications using Large Language Models such as GPT3 .

7 min read · Feb 8

88



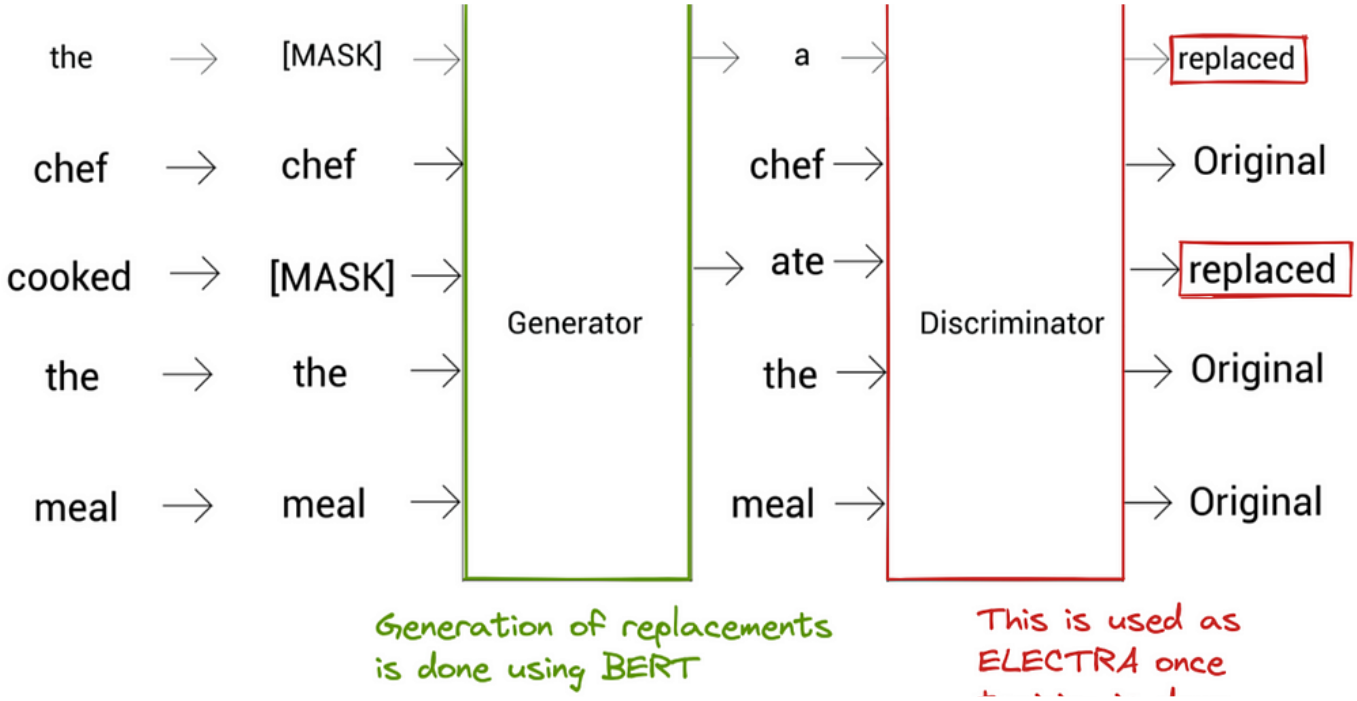
Leone Perdigão

ChatGPT: a deep dive

Unlocking the Potential: How ChatGPT's Exponential Growth and Advanced Architecture is Changing the Game in Natural Language Processing

9 min read · Jan 27

24 1



Harshit Sharma in Towards AI

The Ever-evolving Pre-training Tasks for Language Models

Self-Supervised Learning (SSL) is the backbone of transformer-based pre-trained language models, and this paradigm involves solving...

6 min read · Dec 28, 2022

98 2