Nlp phrase generater

Nlp paraphrase

Search query generator

Nlp Phrase generator

Phrase suggester

Tidbit from searches:

<https://www.tensorflow.org/tutorials/word2vec>

[Vector space models](https://en.wikipedia.org/wiki/Vector_space_model) (VSMs) represent (embed) words in a continuous vector space where semantically similar words are mapped to nearby points ('are embedded nearby each other'). VSMs have a long, rich history in NLP, but all methods depend in some way or another on the[Distributional Hypothesis](https://en.wikipedia.org/wiki/Distributional_semantics#Distributional_Hypothesis), which states that words that appear in the same contexts share semantic meaning. The different approaches that leverage this principle can be divided into two categories: count-based methods (e.g. [Latent Semantic Analysis](https://en.wikipedia.org/wiki/Latent_semantic_analysis)), and predictive methods (e.g.[neural probabilistic language models](http://www.scholarpedia.org/article/Neural_net_language_models)).

This distinction is elaborated in much more detail by [Baroni et al.](http://clic.cimec.unitn.it/marco/publications/acl2014/baroni-etal-countpredict-acl2014.pdf), but in a nutshell: Count-based methods compute the statistics of how often some word co-occurs with its neighbor words in a large text corpus, and then map these count-statistics down to a small, dense vector for each word. Predictive models directly try to predict a word from its neighbors in terms of learned small, dense embedding vectors (considered parameters of the model).

<https://www.quora.com/What-are-some-NLP-approaches-to-paraphrasing-text>:

Basically the main approach is to decompose your sentences into small parts and after that randomly combine them into new one. One of the examples I found online is these three sentences which basically mean the same:

* I hate this guy, he is so dumb;
* This guy is stupid, I hate him;
* I despise this dumb guy.

Here’s a nice topic for you to check out if you want to learn more about it: [What's a good nat...](http://stackoverflow.com/questions/25332/whats-a-good-natural-language-library-to-use-for-paraphrasing)[(more)](https://www.quora.com/What-are-some-NLP-approaches-to-paraphrasing-text)

<https://stackoverflow.com/questions/25332/whats-a-good-natural-language-library-to-use-for-paraphrasing>:

first parses sentences of blog posts, extracts the semantic meaning of these posts, and then searches through the space of vocab words which will compositionally create the same semantic meaning, and then pick one that doesn't match the current sentence.

Fastsubs: <https://code.google.com/archive/p/fastsubs/>

<http://www.denizyuret.com/2012/05/fastsubs-efficient-admissible-algorithm.html>

Lexical substitutes have found use in the context of word sense disambiguation, unsupervised part-of-speech induction, paraphrasing, machine translation, and text simplification. Using a statistical language model to find the most likely substitutes in a given context is a successful approach, but the cost of a naive algorithm is proportional to the vocabulary size. This paper presents the Fastsubs algorithm which can efficiently and correctly identify the most likely lexical substitutes for a given context based on a statistical language model without going through most of the vocabulary. The efficiency of Fastsubs makes large scale experiments based on lexical substitutes feasible. For example, it is possible to compute the top 10 substitutes for each one of the 1,173,766 tokens in Penn Treebank in about 6 hours on a typical workstation. The same task would take about 6 days with the naive algorithm. An implementation of the algorithm and a dataset with the top 100 substitutes of each token in the WSJ section of the Penn Treebank are available from the author's website at <http://goo.gl/jzKH0>.

<https://github.com/orenmel/word2parvec>

A toolkit for generating paraphrase vector representations for words in context

<https://levyomer.wordpress.com/2014/04/25/word2vec-explained-deriving-mikolov-et-al-s-negative-sampling-word-embedding-method/>

word2vec explained: deriving mikolov et al.’s negative-sampling word-embedding method