NLP 89-680
Assignment 1: Distributional Similarity

The goal of this assignment is to explore distributional similarities for words, using the output of word2vec word embeddings and the output of a Large Language Model (LLM, specifically ChatGPT). The text below guides you which explorations to perform, and you're asked to submit a report which presents the results for all the specified tasks.

Dense vectors (word2vec based similarities)

In this part of the assignment, we will explore word-similarities induced by a word-embedding algorithm.

Word vectors

Use the `word2vec-google-news-300` pre-trained vectors, from the `gensim` python package.

https://radimrehurek.com/gensim/models/word2vec.html

You can learn about the different available models here:
https://github.com/RaRe-Technologies/gensim-data

After installing the `gensim` package (using `pip install gensim` from the commandline), you can load the model as follows:

```
"""
import gensim.downloader as dl
model = dl.load("word2vec-google-news-300")
# this will take a while on first load as it downloads a 1.6G file.
# later calls will be cached.

# You can now use various methods of the "model" object.

# you can access the vocabulary like so:
vocab = model.index_to_key
"""
```

Code

Write python scripts that may use the `numpy`, `sklearn` (scikit-learn) and `gensim` packages.

Generating lists of the most similar words

Choose 5 words in the vocabulary, and for each of them generate the list of 20 most similar words according to word2vec. You're asked later to analyze these lists in comparison to the ChatGPT output.

Polysemous Words

Polysemous words are words that have several meanings (for example, the word "bank" or the word "bat"). Find three polysemous words (words with at least two different meanings) such that the top-10 neighbors of each word reflect both word meanings, and three polysemous words such that the top-10 neighbors of each word reflect only a single meaning.

1) Which three polysemous words belong in the first group, and what are their neighbors?
2) Which three polysemous words belong in the second group, what are the possible senses for each word, and which of the senses was reflected in the top-10 neighbors?
3) Can you explain why the second group words neighbors reflect only one sense?

Synonyms and Antonyms

Synonyms (מילים נרדפות) are words that share the same meaning. *Antonyms* are words that have an opposite meaning (like "cold" and "hot").

Find a triplet of words (w1, w2, w3) such that all of the following conditions hold:

a) w1 and w2 are synonyms or almost synonyms.
b) w1 and w3 are antonyms.
c) sim(w1,w2) < sim(w1, w3)

sim should be the *cosine similarity* measure.

1) Write your triplet.
2) Can you explain this behavior in which the antonyms are more similar than the synonyms?

The Effect of Different Corpora

In this section, we would like to compare models based on two sources. The first model is based on wikipedia and news text, and the second based on twitter data. For the wikipedia and news model, use the gensim model `glove-wiki-gigaword-200` . For the twitter data, use the gensim model `glove-twitter-200` (“glove” is a different algorithm than word2vec, but its essence is similar).

Find 5 words whose top 10 neighbors based on the news corpus are very similar to their top 10 neighbors based on the twitter corpus.

Find 5 words whose top 10 neighbors based on the news corpus are substantially different from the top 10 neighbors based on the twitter corpus.

1) Which words did you find?
2) For the second case, describe in words the difference in neighbors you observe.
3) What was your strategy (either automatic or manual) for finding these 10 words?

Plotting words in 2D

Dimensionality Reduction

Dimensionality reduction is a technique by which you take n-dimensional data and transform it into m-dimensional data (m < n), while attempting to maintain properties (such as distances) of the original data. Of course, dimensionality reduction is always a lossy process (because there is more information in n-dimension than in m<n dimensions). Yet, it is still useful.

One way of dimensionality reduction is via the PCA algorithm. This algorithm is implemented in the `scikit-learn (sklearn)` python package. You can use it like so:

```
"""
from sklearn import decomposition

pca = decomposition.PCA(n_components=2)
pca.fit(X)      # use a set of vectors to learn the PCA
transformation
Z = pca.transform(Z)  # transform a set of vectors to reduce their
dim
                      # (it is possible that Z=X)
"""
```

Take the first 5000 words in the vocabulary (`model.index_to_key[1:5000]`).
From these, keep only words that end either with “ed” or with “ing” (you should be left with 708 words). From the resulting list of 708 words, create a matrix with 708 rows, where each row is a 300-dim vector for one word, and reduce the dimensionality of the matrix to 2-d (the result is a matrix with 708 rows and 2 columns). Plot the resulting 2-d vectors (you can use the `scatter` plot in the `matplotlib` package) where each vector/word is a dot in a scatter plot, according

to its coordinates. Color points that correspond to words that end with “ed” in blue, and points that correspond to words that end with “ing” in green. Submit the resulting plot. Also submit a discussion of it (are the colors separated? why or why not? is the information included in the model? etc).

Word-similarities in Large Language Model

In this part of the assignment, we will explore word similarities in ChatGPT.

ChatGPT is a product of OpenAI, which is based on a large-language-model, trained on trillions on words, and then fine-tuned to follow human instructions, in a chat-like manner. You can access ChatGPT in the following URL: https://chat.openai.com.

Let’s explore word-similarities in ChatGPT. Please use the 3.5 version.

Related words

For each of the 5 words for which you generated the similarity lists in the word2vec part, ask ChatGPT to produce a list of 20 similar words. Examine qualitatively the outputs of the two methods. How do they compare in terms of accuracy? in terms of diversity? are they all words, or are some multi-word phrases? if there are phrases, can you make it produce only words?
In addition, try identifying different types of similarity that are reflected in the lists, give some examples for each type, and examine whether these types differ between the word2vec output and the ChatGPT output.

Now select two of the words and increase the number of neighbors from 20 to 100, for both ChatGPT and word2vec, what trends do you observe?

Note: the first prompt (the instruction you give ChatGPT) you try may or may not work well, you are encouraged to experiment with different prompts. In your report, write examples for a few prompts that you tried and which prompt worked best for you.

Synonyms and Antonyms

Can you get ChatGPT to reliably produce synonyms? Which prompt did you use?
Can you get ChatGPT to reliably produce antonyms? Which prompt did you use?

Polysemy

How does ChatGPT behave when you ask it for similarities for the polysemous words you found in the word2vec part? In what senses is it similar, and in what senses is it different from word2vec?

Mean Average Precision (MAP) evaluation

Choose as target words two of the 5 words for which you produced similarity lists using the two methods above. Take the lists of 20 most similar words produced for each target word by each method (4 lists, one by each method for each of the two target words; the two similarity lists for a target word may of course have overlapping candidates), and judge the correctness of their similarities manually, as we learned for the MAP evaluation method. Specifically, for each candidate word in the similarity lists, make two judgments:

1. Whether you judge the candidate word to be topically related to the target word.
2. Whether you judge the candidate word to be in the same semantic class as the target word.

Include your manual judgments in the submitted report and compute the Mean Average Precision for each of the two methods - word2vec and ChatGPT. Which is better?

What to submit

Code:

Submit your plotting code in a .zip file containing all the needed scripts, and a README.md[1] file explaining how to run your code. Additionally, include in the .zip file all other scripts used in the assignment, and add a corresponding section to the READM.md file briefly explaining the purpose of each file and how to run it. In total, your README.md file should have two sections, “Plotting Code” and “Additional Scripts”.

Report:
Write a short report containing the following for each of the tasks specified above:

- The required output
- Your answers and requested discussion or analysis

[1] The `.md` extension means `markdown`, which is a textual format extended with some simple markup. It is widely used in, for example, websites like github. You can learn more about `markdown` here: https://www.markdownguide.org/getting-started/ but for the purpose of these assignments, you are not required to use any of its features, your README file can also be just a plain ascii text file, with the `.md` extension. (Using markdown is nice and convenient, though, so why not do it?)

- Qualitative and/or quantitative results where relevant