Practical 1: word2vec

For this practical, you'll be provided with a partially-complete IPython notebook, an interactive web-based Python computing environment that allows us to mix text, code, and interactive plots.

We will be training word2vec models on TED Talk and Wikipedia data, using the word2vec implementation included in the Python package `gensim`. After training the models, we will analyze and visualize the learned embeddings.

Setup and installation

On a lab workstation, clone the practical repository and run the `install-python.sh` shell script in a terminal to install Anaconda with Python 3, and the packages required for this practical.

Run `ipython notebook` in the repository directory and open the `practical.ipynb` notebook in your browser.

Preliminaries

Preprocessing

The code for downloading the dataset and preprocessing it is prewritten to save time. However, it is expected that you'll need to perform such a task in future practicals, given raw data. Read it and make sure you understand it. Often, one uses a library like `nltk` to simplify this task, but we have not done so here and instead opted to use regular expressions via Python's `re` module.

Word frequencies

Make a list of the most common words and their occurence counts. Take a look at the top 40 words. You may want to use the `sklearn.feature_extraction.text` module's `CountVectorizer` class or the `collections` module's `Counter` class.

Take the top 1000 words, and plot a histogram of their counts. The plotting code for an interactive histogram is already given in the notebook.

Handin: show the frequency distribution histogram.

Training Word2Vec

Now that we have a processed list of sentences, let's run the word2vec training. Begin by reading the gensim documentation for word2vec at https://radimrehurek.com/gensim/models/word2vec.html, to figure out how to use the `Word2Vec` class. Learn embeddings in $\mathbb{R}^{100}$ using CBOW

(which is the default). Other options should be default except `min_count=10` so that infrequent words are ignored. The training process should take under half a minute.

If your trained `Word2Vec` instance is called `model_ted`, you should be able to check the vocabulary size using `len(model_ted.vocab)`, which should be around 14427. Try using the `most_similar()` method to return a list of the most similar words to "man" and "computer".

Handin: find a few more words with interesting and/or surprising nearest neighbours.

Handin: find an interesting cluster in the t-SNE plot.

Optional, for enthusiastic students: try manually retrieving two word vectors using the indexing operator as described in gensim's documentation, then computer their cosine distances (recall it is defined as $d(x, y) = \frac{\langle x,y \rangle}{\|x\|\|y\|}$). You may be interested in `np.dot()` and `np.linalg.norm()`, see the numpy documentation for details. Compare this to the distance computed by gensim's functions.

Comparison to vectors trained on WikiText-2 data

We have provided downloading/preprocessing code (similar to the previous code) for the WikiText-2 dataset. The code uses a random subsample of the data so it is comparable in size to the TED Talk data.

Repeat the same analysis as above but on this dataset.

Handin: find a few words with similar nearest neighbours.

Handin: find an interesting cluster in the t-SNE plot.

Handin: Are there any notable differences between the embeddings learned on data compared to those learned on the TED Talk data?

(Optional, for enthusiastic students) Clustering

If you have extra time, try performing a k-means clustering (e.g. using `sklearn.cluster.kmeans`) on the embeddings, tuning the number of clusters until you get interesting or meaningful clusters.

Handin

See the bolded "**Handin:**" parts above. On paper or verbally, show a practical demonstrator your response to these to get signed off.