# Mini-project 2: embeddings + WordNet

**Train** your own word embeddings model using `gensim`, from scratch, on a text corpus of your choice (don't use pre-trained models from `gensim`). Alternatively, you can train them using other tools such as word2vec scripts. The language of the texts in the corpus can be English or a different language (preferably one covered by WordNet).
Choose the parameters (window size, minimum word frequency, …) that you consider best.
Make sure to do proper pre-processing and tokenization on the text before training the embeddings model (i.e. eliminate punctuation, it's your choice if you lowercase words or normalize accented words and diacritics, generally no need to lemmatize them. The smaller the corpus, the more normalization is recommended).

Corpora suggestions:

- download a corpus from NLTK - explore with the downloader interface to browse through all available text corpora (some examples here https://www.nltk.org/howto/corpus.html#plaintext-corpora),
- download a book from https://www.gutenberg.org/ (some books also accessible through `nltk.corpus.guttenberg`)
- the Bible (https://www.biblegateway.com/versions/ / also accessible through `nltk.corpus.genesis`)
- any other large enough corpus

The corpus should be at least 50,000 tokens (the bigger - the better).

Save the generated embeddings space in a file for later use. You can also explore the resulting embeddings by converting your saved model into a text file and loading it into https://projector.tensorflow.org/ .

**Evaluate** the quality of the generated embeddings space using WordNet as a baseline.
Compute the following metrics:

1. Compute WordNet coverage: Percentage of all words in your embedding space which can be found in WordNet. For each word, you will check whether the word or its lemma occurs in a synset in WordNet.
2. Choose by hand a reasonable cosine similarity score threshold that can help you decide whether two words in the embedding space are synonyms (e.g. cosine similarity $> 0.8$). Define a function that decides whether two words are considered synonyms in the generated embedding space, based on this threshold. Then compare with WordNet synonymy, using the following metrics, defined on word pairs:
    a. Precision: measures the ratio of pairs of synonyms according to the word embedding metric which are also actually synonyms according to WordNet:
    $S_{emb} = \{(w_1, w_2) \mid w_1 \text{ and } w_2 \text{ are synonyms in the embedding space}\}$
    $S_{wn} = \{(w_1, w_2) \mid w_1 \text{ and } w_2 \text{ belong to same synset in WordNet}\}$
    $P = \frac{|S_{emb} \cap S_{wn}|}{|S_{emb}|}$
    b. Recall: measures the ratio of pairs of synonyms according WordNet which were found as synonyms in the embeddings space:

$$R = \frac{|S_{emb} \cap S_{wn}|}{|S_{wn}|}$$

c. F1-score: simple harmonic mean between precision and recall. (https://en.wikipedia.org/wiki/F-score)

In order to compute these metrics, select a random sample of at least 1000 words from words in the embedding space vocabulary (exclude stopwords) and generate pairs between them. Make sure words in each pair are distinct ($w_1 \neq w_2$) Compute the metrics on this sample of word pairs (we will assume the results are a good enough approximation of the full vocabulary).
Whenever searching for a word in WordNet, you can consider its lemma. (meaning $S_{wn}$ is more accurately defined as $S_{wn} = \{(w_1, w_2) \mid lemma(w_1) \text{ and } lemma(w_2) \text{ belong to same synset in WordNet}\}$ )

3. Find the words / word pairs which do not fit the coverage, precision and recall criteria:
   a. *coverage errors*: words in the embedding space which are not covered in WordNet
   b. *precision errors*: word pairs which are considered synonyms in the embedding space but do not appear in the same synset in WordNet
   c. *recall errors*: word pairs which are not considered synonyms in the embedding space, but do appear in the same synset in WordNet

**Deliverables** (to send to me before deadline)

- code used to train the embeddings
- code used to evaluate the embeddings space (points 1, 2, 3)
- statistics and results:
  - mention the chosen corpus, its language, and the number of tokens in the corpus
  - report obtained scores in terms of the 4 metrics above (coverage, precision, recall, F1)
  - a few selected examples obtained at point 3 (a, b and c)

You can send the deliverables either as a Google Colab file or .py and .txt files. If you choose the latter, please upload the files in a cloud storage of your own (e.g. Google Drive) and send the links to me by email (rather than email attachments).

Note: if you use Colab and want to upload your own large corpus into Colab's session storage, it might take a long while to upload - you can optimize this by uploading the file as a .zip for example and unzipping it programatically (`!unzip path.txt`)

**Deadline**: 3 weeks (02.05.2022)
This mini-project accounts for **3p.**

**Bonus points** (+0.5p): Experiment with the word similarity threshold used to decide word embedding synonymy (select a sample of different values) and find the optimal one that maximizes the F1-score computed above.