COM6115: Lab Class 8

Word Embeddings

In this lab you will work with **Gensim**[1], a library for Python that facilitates the training and use of word embedding models. The lab brief is composed of two parts, one dedicated to exploring Gensim and the generated word vectors, and the other to performing text classification (sentiment analysis) using a word embedding model.

For both parts, we recommend the use of Google Colab, by following the link below:
https://colab.research.google.com/drive/1PSKNueUC0hVouzbqXfoD-LZa9zbSCSKYscrollTo=4fO6hUrM5sT7

PART 1: Computing word embeddings using Gensim

For this part, we will use data from the Gutenberg project[2], accessed through NLTK[3]. In particular, we will use Moby Dick by Herman Melville (`melville-moby_dick.txt`). This data will be parsed into sentences and used to train a word2vec model from scratch.

The code for training the word2vec model is given and you will see that it is done in one single line. The lab sheet will then contain code for you to explore the trained embeddings.

Vector Operations

This task presents the idea of operations within the semantic vectors. There is code available showing you how to do it. You are also given a method to calculate cosine similarity between vectors that you can use to assess how similar two vectors are (e.g. how similar are the vectors for 'whale' and 'ship'). You are also given code for plotting the vectors.

In the final part of this task, you are presented with loading a pre-trained model using Gensim. You will then use a pre-trained word2vec model that follows the same configuration as Mikolov et al. (2013). With this larger model, you should be able to check some semantic vector operations that we presented during our lecture (e.g. 'king' - 'man' + 'woman' $\approx$'queen').

Most Similar Method

You are asked to write a "most similar" method that can retrieve the top n most similar words to a given word. You can use the given cosine similarity method to perform this task. The output should be a list of tuples (word, similarity). For instance, for the word 'ship' and $n = 5$, you should retrieve:

[('boat', 0.9945816397666931),
('head', 0.9881510138511658),
('boiling', 0.982123076915741),
('pequod', 0.9806775450706482),
('sperm', 0.9793713688850403)]

[1] https://radimrehurek.com/gensim/index.html
[2] https://www.gutenberg.org/
[3] https://www.nltk.org/book/ch02.html

You will then compare your method with Gensim's pre-built `most_similar` method. Do you get the same results? Which one is faster?

PART 2: Text classification with embeddings

In this part, you will use word embeddings as features for a text classifier. We will apply this for Sentiment Analysis.

Data description

The dataset is a corpus of tweets about US airline companies (a smaller version of the dataset used for our Sentiment Analysis lab). This dataset is a comma-separated file.

- Each sample is composed of a tweet id (first column), followed by the sentiment and the text of the tweet.
- The text has not been tokenized nor lowercased (you might want to preprocess it before use).

The sentiments are expressed on a 3-value scale: positive, neutral and negative. In the following table you can find several example sentences and their sentiment value.

570270684619923457	I❤flying @VirginAmerica.	positive
569347934866637345	@SouthwestAir are you hiring for flight attendants right now?	neutral
569960080734490624	@united I'm rebooked now, but the line was 300 people deep.	negative

Word embeddings model

For this task, you will use a GloVe ("Global Vectors for Word Representation") model[4] (Pennington et al., 2014). This model is an "evolution" of word2vec, where a word-word co-occurrence matrix is used, instead of context windows.

You will use a pre-trained GloVe model trained on a 2-billion item Twitter dataset. You will see how to load this model by downloading it from a server and loading it using Gensim.

Sentiment Classification

Task 1: pre-processing

Before training the GloVe model, a pre-processing step was applied where hashtags, mentions and other textual features typical of Twitter (and other social media) data were replaced by

[4] https://nlp.stanford.edu/projects/glove/

unique tags. E.g. all hashtags were replaced by <HASHTAG>, all mentions by <USER>, all URLs by <URL> and so on. You can find the regular expressions used in this ruby file.

You are then asked to create a function that performs this kind of pre-processing on the airlines dataset.

Task 2: extract sentence embeddings

So far, we have vectors for words, but we need to have a single vector that can represent our tweets. We will derive this from the vectors for the individual words within the tweet. This process is called **pooling**.

The most basic operations are mean, max or min, where you can use the mean, max or min vectors of all words in a sentence to represent the sentence. For example, for the sentence "This is a sentence ." and the mean pooling you would get the vectors for each token, sum them and divide the resulting vector by the total number of tokens. You are asked to implement a mean pooling method.

After extracting the sentence embeddings for all tweets in our dataset, we can then split it into training and test sets and apply our favourite classification model. In this lab sheet, we used SVC (a support vector machine - SVM - implementation for classification) available in the scikit learn[5] library. You can change the classifier and use, for instance, a Naïve Bayes[6] classifier.

Task 3: error analysis

Perform error analysis. Analyse the tweets that were incorrectly classified by your model. Explore the types of mistakes your classifier made. How many negative tweets did it predict as positive? and neutral?

Notes and comments

- To run a notebook, you need to install jupyter https://jupyter.org:
  - with conda: `conda install -c conda-forge notebook`
  - with pip: `pip install notebook`
  - then run the command `jupyter notebook` in your terminal.
- Alternatively, use Google colab: https://colab.research.google.com/drive/1PSKNueUC0hVouzbqXfoD-LZa9zbSCSKYscrollTo=UgK9tfNA1KPe
- Go to Files (icon on the left hand side) → Upload the file `tweets_short.csv`

References

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, Jeffrey Dean (2013): Distributed Representations of Words and Phrases and their Compositionality. Advances in Neural

[5] https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
[6] https://scikit-learn.org/stable/modules/naive_bayes.html

Information Processing Systems, volume 26.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global Vectors for Word Representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.