F21NL (2025-26) – Coursework 1 [10%]

Summary

In this coursework you will explore learning word representations from a large portion of English Wikipedia (WikiText-103; Merity et al., 2016), using predict-based (Word2Vec; Mikolov et al., 2013) algorithms. To evaluate their performance, you will perform the "downstream" task of **Semantic Relatedness** of two words (e.g., how close is the word "doctor" to "nurse"?). You will also experiment with the task of (b) **Word Analogy** (e.g., "*apple is to tree as grape is to* _ *?*", where the missing word should be "*vine*". This coursework is largely inspired by this seminal NLP paper by Baroni et al, 2014, and Wang et al., 2019.

Deliverables: You will NOT have to submit a document report in .pdf or .doc but instead you will enter your responses and upload your code by exporting your work from Google Colab to a Jupyter notebook in the corresponding Assignment created on Canvas. Read the instructions in the last Section 'Deliverables' below for more details.

Note 1: You will be using the Gensim library for the implementations of the algorithms, i.e., you are not going to be required to implement them from scratch. However, as is the case with most NLP projects, you will spend a considerable time *preparing* the training and evaluation data, and of course *training* your models! Along with the instructions of the coursework (this document) you should use this Google Colab (link) which provides some initial code to download the corpus you will need to use for training.

Note 2: A good NLP-er never stops at training a few models and running some automatic evaluations. Two main assessed learning outcomes are going to be: a) good reporting and discussion of results accompanied by b) some intuitions and (at least a best-effort attempt at an) interpretation of why models behave in certain ways.

Part I: Train Predict-Based models and Evaluate [40%]

Step 1 (Environment Setup): First, make a copy of the Google Colab we shared with you in your Google Drive and execute the first 3 cells. These will install all necessary dependencies, download, and unzip the corpus on the remote storage (in fact it will create two different flavours for you, `wikitext_small`, `wikitext_large`). Then load the corpora using the method `loadDataset()` from the 4th cell (we are using the `datasets` library from hugginface; the result is a dictionary; spend some time to familiarise yourself with the returned format). Remember you will have to repeat these steps every time you connect to a VM. Alternatively, you can download the .ipynb Jupyter notebook locally and work on your computer.

Step 2 (Train Baselines): Using `wikitext_small` and `wikitext_large` build two different word2vec models using the `gensim.models.word2vec` class from Gensim (the API can be found here).

Very Important: For each instance of the corpora, you will need to remove stopwords and tokenize at the empty space (“ “) before attempting to train anything.

Tip 1: There are several hyperparameters you can set, such as the number of dimensions (`vector_size`), the number of iterations (`epoch`) , the context window size (`window`) and the minimum number of times each word can appear (`min_count`). You will get the chance to experiment with these values more in Step 9, but we recommend for this exercise to use the following values: `vector_size=50, iter=5, window=5, min_count=5`.

Tip 2: Complete **Steps 3-4** (along with the answers) for `wikitext_small` first, before attempting to use `wikitext_large`, as the latter might take longer to compute.

Step 3 (Semantic Relatedness Task): Download WordSim-353 (Agirre et al., 2009) (link; it should be a .zip file containing several files; you should use the combined.csv file), and import it into your Colab project. Each row of the csv corresponds to a pair of words and a relatedness score on a scale of [0.0-10.0] assigned by human judges. Compute the **cosine similarity** for each pair using their corresponding dense vectors from the word2vec model you built in Step 2.

[Answer directly on Canvas the following question; **20%**]:

a. What are the cosine similarity scores for the following pairs:
   - **plane / car**
   - **planet / sun**
   - **cup / article**
   - **sugar / approach**

Step 4 (Evaluate Semantic Relatedness): Compute the Spearman rank-order correlation coefficient between the predicted cosine similarities and the relatedness scores for the entire WordSim-353 benchmark, using the `scipy.stats.spearmanr` class from SciPy.

Note 1: Make sure you understand what this metric actually computes before using it; you can find more about it in this tutorial.

Note 2: Remember! You will have to repeat this both for `wikitext_small` and `wikitext_large`.

[Answer directly on Canvas the following question; **5%**]:

b. What is the value of the Spearman correlation coefficients computed in Step 4?

The actual Spearman correlation coefficient values can be used to interpret the strength or weakness of the correlation of the variables measured.

[Answer **briefly** on Canvas the following question; **10%**]:

c. How do you interpret each coefficient value with respect to the word similarity task? Are the coefficient values for the two vector space models you created different, and if so, why?

Step 5 (Pretrained word2vec): Although you have trained word embeddings with a rather sophisticated algorithm, even the biggest of the two corpora you used is quite small (~100M tokens) in comparison to what we usually use for large pre-training experiments (up to a few trillion tokens!). For this step use the pretrained Word2Vec vectors `word2vec-google-news-300` (pretrained on ~100B tokens of mostly News articles); use the `gensim.downloader.load` method. Evaluate the model on the Relatedness Score task.

[Answer **briefly** on Canvas the following question; **5%**]:

d. What is the value of the Spearman correlation coefficient and how do you interpret it?

Part II: Hyperparameter tuning and Discussion of Results [40%]

So far, we have been evaluating several *baseline* models on the Semantic Relatedness downstream task. This is usually the first step before putting everything together in a large table of results for comparison and discussion.

Step 6 (Hyperparameter tuning; Longest-step Warning!): Go back to the baseline models you made in Step 2 and run **at least two more** training experiments for each dataset *with the aim to improve your existing coefficient scores.*

Note: It is very important to make sure comparisons are fair, i.e., any experiments and claims made are based on the exact same benchmark dataset (in this case we have two), same training corpus, comparable number of training examples, model parameters, vocabulary sizes and so on.

[Answer directly on Canvas the following question; **25%**]:

e. Create a table of results that summarises your experiments. **Tips**: In case you have run several experiments with different hyperparameters choose to present the most significant. It might be worth presenting more than one experiment with only a single hyperparameter change if you want to emphasise a striking difference worth discussing. Finally, apart from the Spearman correlation coefficients, make sure you also include the most significant hyperparameters as separate columns (for example, see Table 2 from Merity et al., 2016).

[Write a **brief** response on Canvas discussing the following questions; **25%**]:

f. Using the table of results from your answer to question g., write a short discussion section that answers the following questions:
    i. Does a bigger corpus yield better representations?
    ii. Does a bigger vocabulary yield better representations?
    iii. Do bigger word vectors yield better representations?
    iv. Does a bigger context window yield better representations?
    v. Step 6 requires you to *look for* the best combination of hyperparameters using the same two datasets for evaluation. Is this a good practice?

Part III: Word Embeddings Exploration: Analogies [10%]

As we have seen in class, vector representations of words allow us to compute analogical relations of the form *"apple is to tree as grape is to [vine]"* using the parallelogram model (Rumelhart and Abrahamson, 1973).

Step 7 (Analogies): Using the pretrained word2vec vectors from Step 5 implement the parallelogram model to return not just the most similar analogy but the top-5 analogous words.

[Answer **briefly** on Canvas the following question; **5%**]:

g. What are the top-5 analogies for the following configurations:

- **man is to woman as king is to ___?**
- **Athens is to Greece as Rome is to ___?**
- **reading is to read as playing is to ___?**
- **Greece is to souvlaki as Italy is to ___?**
- **airplane is to propeller as car is to ___?**

Is the top-1 answer always the "correct"? What about the rest of the results?

Unfortunately, machine learning algorithms such as word2vec run the risk of amplifying biases in the data, especially when the training corpus originates from very large web-based unfiltered sources. If you want to learn more about this have a look at Bolukbasi et al., 2016. Let's test whether the pretrained word2vec model contains biases, such as gender stereotypes.

[Answer **briefly** on Canvas the following question; **5%**]:

h. What are the top-5 analogies for the following configurations? Can you identify any gender-based stereotypes? Briefly discuss your findings:
   - **man is to woman as computer programmer is to ___?**
   - **man is to woman as superstar is to ___?**
   - **man is to woman as guitarist is to ___?**
   - **man is to woman as boss is to ___?**

Deliverables

You will have to submit the Assignment in ‘Assignments>Coursework 1 on Canvas. **Only one person is required to submit**. Over there, you should (i) upload the exported .ipynb Jupyter Notebook from Google Colab (it should be named: Coursework_1_H0XXXXXXX-H0YYYYYYY.ipynb, by replacing the Xs and Ys with your Student ID numbers), (ii) enter your answers to the questions outlined in the description above.

It is advisable to properly comment your code. **Note**: Any 3rd party source used must be properly cited in the code (see also below).

Important Notes!

The assignment counts for 10% of the course assessment.

You are permitted to discuss the coursework with your classmates and of course with the entire teaching team. However, coursework reports must be written in your own words and the accompanied code must be your own. If some text or code in the coursework has been taken from other sources, these sources must be properly referenced. In particular, for pieces of code you get from the web (e.g., from StackOverflow), minimally you should provide the link where you found it as an inline comment in the Jupyter notebook. Failure to reference work that has been obtained from other sources or to copy the words and/or code of another student is **plagiarism** and if detected, this will be reported to the School's Discipline Committee. If a student is found guilty of plagiarism, the penalty could involve voiding the course.

You should **never give** hard or soft copies of your coursework report or code to another student. You must always refuse any request from another student for a copy of your report and/or code. Sharing a coursework report and/or code with another student is **collusion**, and if detected, this will be reported to the School's Discipline Committee. If found guilty of collusion, the penalty could involve voiding the course.

Pay special attention to all the **Labs** as they should provide lots of insight as to how to tackle most of the questions in this coursework. Also, **CHECK THE DOCUMENTATION OF Gensim** when you are in doubt.

Your assignment should be completed by **15:30 on Monday 20th October 2025**. No individual extensions are permitted under any circumstances. Students who submit after the deadline but within 5 working days of the deadline will be award 0.7x(awarded coursework mark). Submissions that are more than 5 days late will receive 0 marks.

You will receive the final mark, answers, sample code solution and cohort-wide feedback no later than 15 working days.