COMPSCI 4NL3 Homework 3: Word Embeddings

Winter 2026
Due: March 11th

1 Overview

The goal of this assignment is to explore word embeddings, their potential uses and harms.

2 Requirements

You should perform the following steps:

1. Train word embeddings on Simple English Wikipedia
2. Compare these to other pretrained embeddings
3. Quantify biases that exist in these word embeddings
4. Use your word embeddings as features in a simple text classifier
5. Summarize results and lessons learned

See below for details.

3 Dataset

For this assignment, the dataset that you need is already available in an easy-to-use format on the HuggingFace datasets hub here. You can fit an impressive amount of information in relatively little space when using text, as opposed to other forms of information (audio, video, etc.), and the full English Wikipedia is about 20GB. This is still a considerable amount of data, so for this project we will use Simple English Wikipedia. This is a version of Wikipedia that uses simpler English that is easier to understand, and only has about 260k articles, whereas the full Wikipedia has around 7 million. The approaches for this assignment can be scaled up with more time and resources. You may choose to use the full Wikipedia or a similarly large corpus. To load the dataset in Python, you need to install the following packages:

```
pip install datasets
pip install apache_beam
```

and run the following lines of code:

```
from datasets import load_dataset
dataset = load_dataset("wikipedia", "20220301.simple")
# check the first example of the training portion of the dataset:
print(dataset["train"][0])
```

It may also be helpful to apply some text normalization/preprocessing on the data before completing the next steps. This is optional and you can use any libraries you would like for preprocessing, or code you wrote for the previous assignment.

4 Word Embeddings

Next you will train a word2vec model to learn word embeddings and analyze the results.

4.1 Training Word Embeddings

You now have your dataset and are ready to learn word embeddings. I recommend using gensim for learning the embeddings (fasttext is also a good option and there may be other acceptable libraries, if you're not sure, ask on Teams). The goal is to create skip-gram or continuous bag-of-words (CBoW) embeddings from your dataset (the library you use should be able to do these). You do not need to implement the entire training procedure by yourself and may use libraries to assist you.

You should train at least 2 variations of word embeddings. Each set of embeddings should have a d-dimensional vector for each word in your vocabulary. If you are using Google Colab, you should download these embeddings, so you do not have to regenerate them each time you connect. Some variations to consider:

1. skip-gram vs CBoW
2. context window size
3. vocabulary size
4. dimension of embeddings
5. another option you think will be interesting to explore

4.2 Comparing Word Embeddings

You now need to decide on (at least) 5 simple queries to use to query your embedding space. Each query uses vector arithmetic. Your queries cannot all be single word queries. Examples of queries include:

1. king - man + woman
2. Alberta - rose + Ontario
3. frog + shell

You cannot use the examples above in your report. For each query, you should report the 10 most similar words you get as a result. You should run each query on each set of embeddings you trained, as well as two sets of pretrained embeddings from the following list:

1. GloVe
2. word2vec demo embeddings
3. fastText vectors
4. Any gensim word2vec, GloVe, or fasttext embeddings from here
5. Urban Dictionary Embeddings

You may use other embeddings you find. You cannot use contextual embeddings. The pretrained embeddings must be static (not from ELMo, BERT, or any LLM or transformer-based method). The results of these queries will go into your report.

4.3 Bias in Word Embeddings

Even though word embeddings are trained in an unsupervised way, they are certainly not objective measures, and in fact, are good at capturing potentially harmful stereotypes. To better understand this, see the example at the bottom of the first page of this foundational paper on bias in embeddings. You can try something like this with your own embeddings. If you want to use multi-word phrases, you may want to experiment with a preprocessing step that merges words with high PPMI (over a certain threshold). The *Phrases* gensim package can help with this. Many types of bias that we see in static word embeddings are still present in more recent LLMs, e.g. ChatGPT.

Since the initial bias paper from 2016, several other metrics to measure bias in word embeddings have been proposed, and many of them are generalized in the Word Embeddings Fairness Evaluation Framework (WEFE). There are two example study replications shown here with their code. Choose either the WEAT or RNSB example from that page, read about what it is doing, and think about one extension that you can make. For example, for WEAT, what is a new set of word lists that you could test that would explore a different type of bias? Or for RNSB, what is a different list of words/identities that should not have any inherent differences in their sentiment? Add your extension and rerun the full example using the four sets of embeddings that you compared in the previous step. You do not need to generate the exact same types of figures in the examples, but you can include them in your report if you wish.

A tip for getting the WEFE framework to work correctly if you are trying to reproduce the WEAT example, you may need to update the call of `run_queries()` to something like this:

```
wefe_results = run_queries(WEAT, queries, models, metric_params={
               'preprocessors': [{}, {'lowercase': True}]},
               warn_not_found_words=True
               ).T.round(2)
```

The way it is written in the example `.ipynb` file was using a deprecated argument for that function, which makes the preprocessing work incorrectly if you are using the latest version of `wefe`.

4.4 Text Classification

Now that you have seen some of the problems with word embeddings, let's take a look at how useful they can be as simple features for our models. Train a simple logistic regression classifier for any text classification task. You may, for example, use the training data from the previous assignment. You should compare to a classifier with bag-of-words features (you may directly reuse the code and data from the previous assignment if you wish). Evaluate your model on a held-out test set and report the accuracy and F1-scores. Train a second model, but instead use one of your word embedding models to get features (you may also use the pretrained embeddings). For each document, average together all embeddings for the individual words in that document to get a new, d-dimensional representation of that document. Note that the number of input features is now d, instead of the size of your entire vocabulary. Compare the results of training a model using these features (averaged embeddings are sometimes called *mean pooled* embeddings) to your bag-of-words model. Consider the efficiency of the models when writing your discussion.

5 Deliverables

You should submit the code you used as well as a PDF report documenting your approach and findings.

5.1 Code

Your code should be written in Python and should include enough documentation/instructions for someone else to be able to run. You must submit your code via Avenue. You will have to use the Jupyter notebook template provided to you and fill in the sections as necessary. Your code should run using Python 3.14 and make sure your file read/write operations use UTF-8 (this makes it much easier for us to run your code).

You are welcome to use code snippets from examples in class, things you find online, or from AI code generation tools. Just make sure to give proper attribution to code you did not write. Follow the syllabus instructions for how to report the use of AI tools. This includes reporting estimated emissions! However, you may not copy code that does the entire assignment (e.g. someone who did this assignment in a previous semester).

You do not have to automate the entire process from dataset collection to generation of the figures and tables for the report. You may, for instance, generate the figure using another tool like Libre Office Calc. Your code should showcase how you did things like calculating and comparing embeddings.

5.2 Report

In place of the report, you should answer questions about your work in the provided template notebook. These questions cover the following:

1. **Embeddings.** Describe two variations of embeddings that you trained, which libraries you used to train them, how they differ (in terms of the hyperparameters/settings, not results yet) and what you learned from the process of training these yourself. Then, present the results of your five queries for the two models you trained and the two pretrained models you downloaded, and describe what you found interesting or surprising about these results. Make note of anything you found to be significantly different between the embeddings.

2. **Bias.** Describe which bias study you extended, how you extended it, and what you observed. Present the results of the replication including your extension. Reflect on, and answer the following question: What do you think the consequences of using these embeddings might be when using them as features in a machine learning model?

3. **Classification.** Present the results of your classification experiments. Describe which embedding model you selected, which classification task you used, and why you think you achieved the results that you did.

4. **Reflection.** What did you learn during the completion of the assignment? What was unexpected? What challenges did you face and how did you overcome them?

The answers/results should be well-organized and professionally presented. Please avoid things like blurry, low-resolution or poorly-cropped screenshots, submitting one long paragraph with no subsections or formatting, or printing long strings from your program output with no formatting applied.

6 Double Check Your Files

Immediately after submitting your assignment, go back to the submission page and download your work. Make sure that you can open it, that the files open properly, and that you have submitted the proper files. You must make sure that your PDF can be opened and is not corrupted, encrypted, or otherwise damaged. When it comes time for grading, if you submitted the wrong files, there is no way to prove that you completed the work on time and you will receive a zero for whatever aspect of the assignment is missing.

7 Help

Please use the Teams channel to ask questions about the assignment when you need guidance or pointers on this homework. You are free to discuss your approach and ideas with classmates, but should not share code or reuse data. You may use generative AI tools if you find them helpful, but please clearly document how they were used in the report and follow the guidelines in the syllabus for what you **must** include when using generative AI. If you use generative AI and do not report it, you may receive a 0 for the assignment. You take full responsibility for the deliverables you submit.