Natural Language Processing – Assignment I

Reykjavik University – School of Computer Science

Fall 2025

The purpose of this assignment is to experiment with several text processing tools, for example in the context of tokenization, exploring word embeddings, and writing Python programs. The assignment consists of 4 parts, in total worth 64 points.

1 Python: Gutenberg corpus (16p)

Write a Python program, *corpusAnalysis.py*, which prints out various statistics/information about a given text file, which is a part of the Gutenberg corpus (accessible using NLTK). The name of the text file is given as a parameter to the program, i.e.

```
python corpusAnalysis.py carroll-alice.txt
```

The program should print out the following information for the given text file:

```
Text:  carroll-alice.txt
Tokens:  34110
Types:  3016
Types excluding stop words:  2872
10 most common tokens:  [(',', 1993), ("'", 1731), ('the', 1527), ('and', 802),
('.', 764), ('to', 725), ('a', 615), ('I', 543), ('it', 527), ('she', 509)]
Long types:  ['affectionately', 'contemptuously', 'disappointment', 'Multiplication']
Nouns ending in 'ation'  ['usurpation', 'station', 'accusation', 'invitation',
'consultation', 'sensation', 'explanation', 'Multiplication', 'conversation', 'Uglification',
'exclamation']
```

Note: Long types are those with more than 13 characters.

Return your program code (.py file) along with the output of your program when running against the file *austen-emma.txt*.

2 Tokenization (16p)

In this part, you explore two common methods for tokenizing text: WordPiece and Unigram/SentencePiece.

Use the following Colab notebooks that implement each method to carry out the tasks below (sourced from the Huggingface LLM course[1]): WordPiece and Unigram/SentencePiece.

- Choose a small corpus to build your tokenizer. This could be a collection of poems, short stories, news articles, or any other text your choice in any language.
- Write a paragraph explaining why you chose this corpus. Is it useful for some particular task?

Write at least a paragraph in response to each of the following questions:

[1] https://huggingface.co/learn/llm-course/en/chapter0/1

1. (4p) What is the fundamental difference between each tokenization method?

2. (4p) In the WordPiece script, what does the `vocab_size` variable do? What effect does it have on the output and the runtime to increase or decrease it by a factor of 10?

3. (4p) In the Unigram/SentencePiece script, what does the `percent_to_remove` control? What happens if you change that value? Answer in general and for your corpus in particular.

4. (4p) Which tokenization method was the most useful for your corpus and why?

3 NLTK: The Icelandic Gold Standard – 16 pts

In this part, you use Python and NLTK to process a Part-of-Speech tagged Icelandic corpus, the *Icelandic Gold Standard; MIM-GOLD*[2]. A preprocessed version of MIM-GOLD containing one sentence per line (with "/" between tokens and tags) is available as the file *MIM-GOLD.sent* in the zip file linked the assignment description.[3]

This part is divided into the subparts described below, but you should return a single Python program, **mim_gold.py**, which includes all the code needed for carrying out these tasks.

Write code to:

1. Read in MIM-GOLD using the class *TaggedCorpusReader*, display the number of sentences, and display the individual tokens of sentence no. 100.

2. Display the number of tokens and the number of types in MIM-GOLD.

3. Display the 10 most frequent tokens in MIM-GOLD using the class *FreqDist*.

4. Display the 20 most frequent tags in MIM-GOLD using the class *FreqDist*.

5. Generate tag bigrams and use the class *ConditionalFreqDist* to print out the 10 most frequent tags that can follow the tag 'af' (this tag denotes a preposition).

Example output follows:

```
Number of sentences: 58412
Sentence no. 100:
Púttað á Listatúni í dag , laugardag , kl. 10.30 .

Number of tokens: 1000218
Number of types: 106529

The 10 most frequent tokens
. => 49066
að => 35749
og => 33813
, => 29990
í => 27622
á => 21833
er => 16604
sem => 15199
til => 9888
um => 8799

The 20 most frequent PoS tags:
AF => 109899
```

[2] See `https://clarin.is/en/resources/gold/`

[3] Please make sure not to distribute the MIM-GOLD corpus.

```
AA => 74151
C => 68278
PL => 53212
SFG3EN => 36289
PK => 30647
SNG => 26345
TA => 22992
SFG3EÞ => 19773
CN => 19540
SÞGHEN => 17190
PA => 13043
N----S => 12783
CT => 12664
SFG3FN => 12294
NKEN => 11411
NVEN => 11171
NVEO => 11123
NHEÞ => 10979
NVEÞ => 10281

The 10 most frequent PoS tags following the tag 'af':
NVEÞ => 5714
NHEÞ => 5361
NKEÞ => 4222
CN => 3903
NVEO => 3556
NKEO => 3347
NHEO => 3046
NHEÞG => 2867
NVEÞG => 2723
FPHEÞ => 2703
```

4 Bias in Word Embeddings – 16 pts

Word embeddings are dense vector representations of words that have successfully improved the state-of-the-art on many NLP tasks. These vectors are produced by training classifiers on text corpora, such as Wikipedia, Twitter, and news sites. This data may contain biases of various kinds, e.g., denigration, stereotyping, recognition, and under-representation. Specifically, word embeddings may contain gender bias [1, 2], sexual orientation bias [3], ethnic bias [4], and ageism 5. These biases may be present in the word embeddings themselves and therefore have an adverse effect on the NLP task they're being used for, e.g., text classification, text summarization, language generation, and machine translation [6].

Investigate whether there are any potential biases to be found in word embeddings created using (1) data from Wikipedia and (2) data from Twitter (e.g. `glove-wiki-gigaword-100` and `glove-twitter-100`). There are several methods you can use to accomplish this task. For example, Lab 3 has methods for finding similarities and relationships between words that you can use and adapt, as you see fit (e.g., *find_word*, *most_similar*, *similarity*, *similar_by_vector*, *doesnt_match* – See more in the Gensim docs, like KeyedVectors). You may for instance explore combining vectors in various ways (e.g., using the NumPy package) or use any other method you discover.

Answer the following questions in essay-form:

1. (4p) What kinds of biases are present in either dataset, if any?
2. (4p) Are there any differences with respect to bias between the two kinds of data?

3. (4p) Can biases between concepts be revealed through visualizations?

4. (4p) Can bias in word embeddings be removed? If so, how?

Write about your findings and show example code you used to investigate this, including examples of relevant output. Additionally, there are several web-apps on the internet for visualizing word embeddings (e.g., WebVectors). Use one of these tools to plot relations between concepts and include in your hand-in.

5 What to return

A single .zip file containing the following:

1. The Python program **corpusAnalysis.py** and a file containing the output of your program when running against the file *austen-emma.txt*.

2. The Python program **mim_gold.py**. Make sure you use functions (where appropriate) for specific tasks in your Python code, thus minimizing the duplication of code.

3. A PDF file that contains: (*i*) the responses to questions 1-4 in Section 2 and (*ii*) the essay in Section 4.