Lab 1

Welcome to the first lab! Starting off, please familiarize yourself with the Github repository (https://github.com/tollefj/TDT4310-spring-2023). You will need to clone it to be able to work on the labs.

All exercises/labs will follow this format: - A markdown file (also supplied as a PDF) with questions relevant to the current lab - Server code that you will implement to improve the keyboard, found under `labs/your_implementations/lab_n.py`.

It is highly recommended to complete the initial questions before moving on to the implementation.

1) Tokenization

Consider the following sentence:

> *My first car (1974 Ford Pinto) was a trash-can on wheels…*

1. How would *you* tokenize this sentence into words? No coding required (nor is there a "correct" answer)
2. If you were to type the sentence on your phone, what would you expect the next prediction to be after typing "My first car"?
3. What are some of your initial thoughts on the difficulty of next-word prediction in general?

2) Introduction to language modeling

Language modeling is, in short, the task of predicting the next word in a sentence. You may have heard of BERT, GPT, and other language models. A simpler language model can implemented using **n-grams**. You will be implementing a *bigram* and *trigram* model in this lab. Before you tackle this task, some basic knowledge might come in handy.

With the sentence above, use NLTK to get:

1. Bigrams
2. Trigrams

We can continue the n-gram model infinitely (4-grams, 5-grams, …)

3. Explain why we should avoid too large values for n
4. What do you think happens when we select a large n value for a small corpus?

3) Introduction to word representations

Word representations, or word embeddings, are a way to represent words as vectors. Word vectors will be described in greater detail later in the course.

For now, we'll consider a very simple implementation based on the bag-of-words representation.

With the sentence:

> *That that is is that that is not. Is that it? It is.*

- NLTK allows us to create the following frequency distribution using `FreqDist`:
  - `{'is': 6, 'that': 5, 'not': 2, 'it': 2}`
- By considering the words as their indices in the vocabulary...
  - *that* → *0*, *is* → *1*, *not* → *2*, *it* → *3*
- The sentence can be represented as the values of each index:
  - `[5, 6, 2, 2]`

Answer the following:

1. What are your thoughts on the usefulness of this representation?
2. How could you use this technique to compare sentences with each other?