Lab 2

Deadline: February 10, 2023

By now, you should be familiar with the very basics of language modeling and the processeses surrounding the task – such as preprocessing, tokenization and word representations. In this lab, we'll focus more on what features or values we can extract from words, by using (still rather basic) techniques.

Keywords: - Stemming, TF-IDF, part-of-speech tagging and lemmatization

You will deliver the following

A zipped file with your NTNU user name, containing two files:

- A document (any sensible format, but PDF preferred) which answers the tasks in this document
- A copy of your `lab_2.py` file at `labs/your_implementations/lab_2.py`

Setup

You will be using spaCy, which contains slightly more advanced features than NLTK, and also provides pretrained models for several languages. Please take a look at this url: https://spacy.io/usage/linguistic-feature

Installation guide:

```
pip install -U spacy
python -m spacy download en_core_web_sm
```

*sm** indicates the *small* english model. Feel free to install the larger ones, which will be more accurate.

Example usage:

```
import spacy
nlp = spacy.load("en_core_web_sm")
doc = nlp("nlp is fun!")
for token in doc:
    explanation = spacy.explain(token.pos_)
    print(token.text, token.lemma_, token.is_alpha, token.is_stop, token.pos_, end=" ")
    print(f"({explanation})")
```

→

```
nlp nlp True False PROPN (proper noun)
is be True True AUX (auxiliary)
fun fun True False ADJ (adjective)
! ! False False PUNCT (punctuation)
```

In this example above, you should be familiar with things like `is_stop` (whether the token is a stopword) and `is_alpha` (whether the token is a word).

1) Text normalization

Text normalization deals with a large set of techniques, but most commonly:

- Dealing with unwanted characters and symbols
- Lower/uppercasing
- Stopword filtering
- Stemming and lemmatization

The last point is the task of converting text to its standard form. While stemming is a simpler process of reducing words to a common core (*stem*), often by removing suffixes, lemmatization will use dictionaries and definitions to convert words to their base form. The differences can be illustrated as follows:

Word	Stemming	Lemmatization
information	inform	information
informative	inform	informative
computers	comput	computer
feet	feet	foot

1. Given the current state of your smart keyboard (from Lab 1), how do you think you could use the above to improve upon the prediction task? Also discuss why it may fail to give us the words we expect.

2) TF-IDF

TF-IDF (term frequency-inverse document frequency) is a way to measure the importance of a word in a document.

Consider the following documents:

- *I love cats*
- *I love dogs*
- *I love cats, but I also like dogs*

1. Calculate the TF-IDF for "love" and "like" for the last document.
    - Tip: use NLTK's FreqDist to get the bag-of-words (as seen in the last part of the lab 1 handout). Apply preprocessing you think is necessary.
2. Would you replace words with their TF-IDF values? Why or why not?
3. Why should you use the logarithm of the inverse document frequency?

3) Part-of-speech tagging

Both NLTK and spaCy supports POS-tagging. Let's compare a few different approaches.

Consider the sentence:

> *I saw her duck*

And print out the POS tags by...

1. Training a `UnigramTagger` with a `DefaultTagger` as backoff (defaulting to nouns).
   - See the `nltk.tag` module. The training data should be the `Brown` corpus.
2. Using spaCy built-in tagger
3. Discuss why you think the results differ.

Implementation

For Lab 2, I want you to get more familiar with spaCy, while reusing the models created in Lab 1. As usual, the details are found in TODOs in the python file (`lab2.py`).

A running version of Lab1 will also be published in case you had trouble with your implementation.

Notes

As a final part of the lab, I want you to briefly discuss your approach to the implementation task. This could be things you had to learn, difficult parts of the lab, etc.