Homework 3

Concepts and Applications in NLP

January 20, 2025

1 Working with Python NLTK

NLTK (Natural Language Toolkit) is a tool for building Python programs for Natural Language Processing. This set of exercises will look at different functionalities.

1.1 Working with Text Corpora

NLTK includes a small selection of texts from the Project Gutenberg electronic text archive. You can use `nltk.corpus.gutenberg.fileids()` to see the file identifiers.
More information: section 1 in `https://www.nltk.org/book/ch02.html`

For this exercise, get the corpus `austen-emma`.
You can use `nltk.corpus.gutenberg.sents('austen-emma.txt')` to read the data divided into sentences, where each sentence is a list of words.

(Divided into words: `nltk.corpus.gutenberg.words('austen-emma.txt')` and
raw data: `nltk.corpus.gutenberg.raw('austen-emma.txt')`)

POS-tagging: apply POS tagging to the corpus,
see here: `https://www.nltk.org/api/nltk.tag.pos_tag.html`

Lemmatization: lemmatize your data, ideally using POS-tag information.
See here: `https://www.nltk.org/api/nltk.stem.wordnet.html`

Some statistics: what are the most frequent nouns, verbs and adjectives? Give the top-10 each for word forms and lemmas.

1.2 N-Gram Language Modeling

Go through the documentation for n-gram language modeling:
`https://www.nltk.org/api/nltk.lm.html`

Using `padded_everygram_pipeline`, train a 3-gram model on the *austen-emma* corpus.

What are the counts for *great importance*, *great* and *importance*?
Output the probability for $P(importance|great)$.

Use your language model to generate sentences providing different random seeds. What do you think of the generated output?

1.3 Word Similarity and Word2Vec

In this exercise, we train word embeddings on the *austen-emma* corpus and load a (much better!) pretrained model. Then we use the models for some simple tasks.

See here for the documentation:
`https://www.nltk.org/howto/gensim.html#sample-usage-for-gensim`

Model 1: lowercase the *Emma* corpus and train a model. You can try different parameter settings, for example `gensim.models.Word2Vec(data, vector_size=300, window=5)`

Model 2: train a model on lowercased lemmas instead of words.

Pretrained model: load the pretrained model.

Word similarity: output the cosine similarity for two words (e.g. *daughter–sister* and *house–letter*). Compare the output of the three models.

Most similar words: output the top-3 most similar words for some nouns, verbs and adjectives. Compare the output of the three models.

Semantic regularities: use the `most_similar` function to capture syntactic and semantic regularities of the type *King* - *Man* + *Woman* → *Queen*.
(Note: the *Emma*-based models are not very good, the pretrained model gives much better results.)

Some ideas:

- Family relations: *brother* - *man* + *woman* → *sister*
- *yellow* - *banana* + *cherry* → *red*
- *fly* - *air* + *water* → *swim*
- *read* - *book* + *music* → *listen*
- *summer* - *warm* + *cold* → *winter*

Can you find more pairs? Briefly describe your strategy in searching for them.

1.4 WordNet

Look at the documentation of WordNet: `https://www.nltk.org/howto/wordnet.html`

What are the synonyms of *small*? What are the antonyms of *small*?

Go back to the previous task and use the pretrained w2v-model to obtain the top-10 most similar words for the adjectives listed below and check whether the predictions contain synonyms and/or antonyms of the input word.

Adjectives: *shiny, cold, young, good, cheap, expensive, happy, new, ancient, modern, small, large, simple, round, empty*