CS 445 Natural Language Processing

Project 2: Text Exploration V.3

Due Date: December 10, 23:55

This is the final version of the project. Please make sure to do everything described here.

In this project, you are expected to implement some functions to explore the documents or words of a provided document collection. You will use the provided text collection in order to do the followings:

- See whether the Zipf's Law holds true
- See whether the Heaps' Law holds true
- See the most important terms of the collection or a document with a Word Cloud
- See how to build statistical language models and use them to generate sentences
- See how to build distributed word vector representations and use them to estimate word relationships

Furthermore, you will be given a Turkish data collection in different sizes to test these implementations. You will write your findings, results and interpretations into a report and submit that as well.

Implementation Part:

You will be provided with a ***main.py*** script which contains a sample test case. You should not change this *main.py*. During the automatic grading a similar python script will be used, therefore your implementations should work with the *main.py*. Make sure to download the updated main.py.

You need to do your implementations on ***project02.py***. The *main.py* imports *project02.py*. The *main.py* should work without any problems given that your *project02.py* has all the necessary functions implemented. If the provided *main.py* does not work, then your project is incomplete. At the end of the project, you are supposed to **submit only the *project02.py* not *main.py***.

Here are the functions you need to implement:

- Create_WordCloud:

  This function will take the following parameters as input:
  - List of documents

- The dimension size (same dimension will be used for both height and width of the output figure)
- The full path of the output file
- The term *weighting* option (either TF or TFIDF weighting, default is TF)
- The *stopwords* option (either to remove the stopwords or not, default is to keep them)

The function will create the word cloud and save it in the provided output file in png format.

You can use NLTK's stopword lists and scikit-learn's `TfidfVectorizer`[1] and `CountVectorizer`[2].

- Create_ZipfsPlot:

  This function will take the following parameters as input:
  - List of documents
  - The full path of the output file

  The function will create the Zipf's plot and save it in the provided output file in png format.

- Create_HeapsPlot:

  This function will take the following parameters as input:
  - List of documents
  - The full path of the output file

  The function will create the Heap's plot and save it in the provided output file in png format.

- Create_LanguageModel:

  This function will take the following parameters as input:
  - Type of the model (MLE or KneserNeyInterpolated)
  - List of documents
  - Maximum ngram size

  The function will return the trained language model.

- Generate_Sentence:

  This function will take the following parameters as input:
  - Trained language model
  - Text for starting the sentence

---

[1] https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

[2] https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html#sklearn.feature_extraction.text.CountVectorizer

This function will take your trained language model and starting text for the sentence. Then it will generate a sentence by predicting the next word until the end of the sentence token (</s>) is seen. In this function, you will generate 5 sentences and then you will return the sentence which has the lowest perplexity score. You will return the sentence and its perplexity score.

An Example:

```
--> LM3_MLE = project02.create_LanguageModel(Docs,model_type="MLE",ngram=3)

--> sentence,perplexity = project02.generate_sentence(LM3_MLE,text="milli")

--> print(sentence,perplexity)
--- milli takıma da başarılar dileyen ünal aysal , nişantaşı'ndaki bir mekandan 35 yaşındaki erhan yanar'ın eşi 30 kilometre uzaklıktaki komşu köylerden karşılayan vatandaşlar , dakikalarca sanatçıyı izleyip fotoğraf çekti . 3.3155995797081617
```

- Create_WordVectors:

  This function will take the following parameters as input:
  - List of documents
  - The dimension size of the vectors
  - Type of the model (Skipgram or CBOW)
  - Window size

  The function will return the trained word embeddings.

- Use_WordRelationship:

  This function will take the following parameters as input:
  - Trained distributed word representation model
  - Example Tuple List
  - Example Tuple with missing value

  This function will take your trained word embeddings model, an example list of tuples of words which have the same type of relationship and example tuple with the missing value. In this function, you will use the example tuples to find average distance between pair of words, and then use this distance to predict the missing value in the example tuple.
  Example tuples may include words that are not in your vocabulary. In such a case, you have to exclude these tuples from your list of tuples before starting your calculations. If there are no examples left in the list of tuples or the test tuple word is not in the vocabulary, you have to print the following message “Sorry, this operation cannot be performed!”.
  In the output you will print the top 5 candidates for the missing value as tuple together with its similarity score. In some cases, the test word itself could be in the top-5 candidates list, please make sure you ignore the test word while printing the top-5 candidates.
  An example is provided as follows:

```
>>> example_tuple_list =[('fransa','paris'),
                         ('almanya','berlin'),
                         ('italya','roma'),
                         ('ispanya','madrid'),
                         ('hollanda','amsterdam'),
                         ('ingiltere','londra'),
                         ('türkiye','ankara')]

>>> example_test_tuple =('rusya','')
>>> project02.use_WordRelationship(WE, example_tuple_list,
example_test_tuple)
--- [('moskova', 0.7316931486129761),
     ('ukrayna', 0.6547974944114685),
     ('kırım', 0.6009345054626465),
     ('rus', 0.5904118418693542),
     ('putin', 0.5817922353744507)]
```

We should be able to run your code with the following command:

python main.py document_collection_path

Please do not submit ipynb files. Make sure that your project02.py is human readable. Do not export from ipynb. You can do your developments in there but at the end make sure to copy paste them to a python file for submission. You need to make sure that your python script runs with the above command without any problems

You can use the popular/standard python packages. In case you are not sure of a particular library, please ask the instructor or TA. You will need the `matplotlib.pyplot`[3] library for plotting. For wordcloud you will use `WordCloud`[4] library. For Heaps and Zipf's, you do not need any specific library. For language models, you will use `nltk.lm`[5] module. For word embeddings you will use `gensim`[6] module.

The output file should be in the exact same format described above. We will use automatic scripts to check your outputs. All your submissions will be automatically unzipped, tested and graded. So make sure to abide the project instructions carefully at each step. Otherwise, penalties will be applied.
You are expected to implement this project at your own. Your scripts will be analyzed by using state-of-the-art tools for any type of plagiarism.

Report:

You will be provided with a Turkish news articles dataset. The original dataset is huge. We will provide this dataset in different sizes so that you can use the smaller one during the development for efficiency, and finally use the larger ones for your report.

---

[3] https://matplotlib.org/api/_as_gen/matplotlib.pyplot.html
[4] https://pypi.org/project/wordcloud/
[5] https://www.nltk.org/api/nltk.lm.html
[6] https://pypi.org/project/gensim/

In your report, you will put the Zipf's and Heap's plots and interpret them. Similarly you will put the wordcloud with different weightings, and with and without stopwords, and interpret these. You will create language models with different ngrams (1 to 5) and compare their performances in generating sentences. You will train word embeddings with different approaches, window size and dimensions. You will analyze how these models perform while estimating word relationships.

You are expected to write this report at your own. Your report will be analyzed with Turnitin.

All files should be under the same directory (named as your student ID, only the 5 digit numbers), which you will zip and submit.

Submission Instructions:

- You will submit this homework via SUCourse.
- Please check the slides for the late submission policy.
- You can resubmit your homework (until the deadline) if you need to.
- Please read this document again before submitting.
- After submitting, you should download your submission to a different path to double check whether everything is in order.
- Please do your assignment individually, do not copy from a friend or the Internet. Plagiarized assignments will receive -100.