Assignment 3 - Google Pygram and Gensim Library

TDT4117 - Information Retrieval (Autumn 2024)

DUE: **October 22, 2024 23:59**

Start Here

- **Collaboration policy:**
  - You are expected to comply with the University Policy on Academic Integrity and Plagiarism.
  - You are allowed to talk with other students on homework assignments.
  - You can share ideas but not solutions; you must submit your solutions individually or as a group of two.
  - All submissions will be compared against all works submitted this semester and in previous semesters.
- **Submissions:**
  - Note that if you submit your work multiple times, the most recent submission will be graded.
  - You can do this assignment individually or in a team of two people; if you are in a team, just a single submission from either of the team members on Blackboard is sufficient.
  - When you are done, save the Notebook file in a single .zip file with the filename **'group_no_Assignment3_TDT4117.zip'** and upload the .zip file on BlackBoard via the submission link for the assignment. Also, include the names of team members in the Notebook (or only your name if you are working on the assignment individually).
  - The assignment must be delivered in .zip format. Other formats, such as .py and .ipynb, are not allowed.

  Important: *Post all questions on the Ed platform and only send an email in case of (rare) anomalies!*

Assignment Overview

This assignment consists of two parts:

1. Experimenting using Google Pygram library.
2. Indexing with the Gensim library.

Part 1: Google Pygram Library

Google Pygram is a tool developed for analyzing n-grams (sequences of n words) in a large corpus of text. An n-gram is a contiguous sequence of ’n’ words from a given sample of text. Pygram uses data from Google’s Ngram Viewer, which is based on a massive dataset that tracks the frequency of phrases (n-grams).

A Jupyter notebook has been provided for this task, which guides you through using the PyGram library to experiment with Ngram data. To begin, download the notebook to your local machine and install the necessary libraries.

Task: We will use the Google Pygram library to rank time intervals by analyzing the frequencies of bigrams like "Google *" over a specified period (1980-2019). The focus will be on examining how the relevance of different Google products changes over time. From the initial plot, it becomes clear that periods before 1999 are not relevant, and stop words (e.g., "*", "and", "is", "’s") must be removed. After pre-processing the data, we visualize the results in the notebook.

Plotting and Visualization

Using the retrieved data, we will plot the relevance of various Google products over time. From observation, we can deduce that between 1999 and 2012, Google search was more relevant than Google Map, while from 2012 to 2019, Google Map gained more prominence.

Computational Approach: Automating Relevance Detection

To automate the comparison of relevancy between terms over time, we will use the concept of KL Divergence. KL Divergence measures the difference between two distributions. In this case, we treat each year as a time point x and compare the frequencies of two phrases, $P(x)$ and $G(x)$, where:

- $P(x)$ represents the frequency of the phrase "Google Map" at each time point x,
- $G(x)$ represents the frequency of the phrase "Google search" at each time point x.

The formula for KL Divergence is as follows:

$$D_{KL}(P \parallel G) = \sum_{x \in X} P(x) \log \left(\frac{P(x)}{G(x)} \right)$$

By computing this divergence over time, we can determine the periods when one phrase was more relevant than the other. For example, the calculation shows that from 2013 to 2019, Google Map became more relevant than Google search.

Final Task: Extending the Analysis

You are now required to replicate this analysis for two other Ngrams:

- *"Android *"*
- *"Machine Learning with *"*
- *"Norwegian Computer *"*

For each pair of phrases, follow these steps:

1. Pre-process the data by removing stop words.
2. Select two instances of the Ngrams for comparison.
3. Compute the KL Divergence values across all time points.
4. Analyze the results to determine which term was more relevant during specific periods.

By following this procedure, you will gain insights into how the relevance of these terms fluctuated over time, similar to the analysis conducted for Google Map and Google search.

Part 2: Gensim Library

Gensim is a Python library for topic modelling, document indexing and similarity retrieval mainly used in the natural language processing (NLP) and information retrieval (IR) community. In this assignment, We will index and query the book "An Inquiry into the Nature and Causes of the Wealth of Nations" by Adam Smith. Download the text from Project Gutenberg.

Our final task will be indexing and query paragraphs using LSI (over TF-IDF). For example, the query “Do states accept individual donations?” should return the top 3 relevant paragraphs (up to 5 lines each) from the book according to LSI (over TF-IDF) model (see paragraphs below). Also, plot the frequency distribution of the top 15 words of the preprocessed data.

```
[Paragraph 187]
While we cannot and do not solicit contributions from states where we have not met the solicitation requirements, we know of no prohibition against accepting unsolicited donations from donors in such states who approach us with offers to donate.


[Paragraph 92]
CHAPTER II.
OF THE DISCOURAGEMENT OF AGRICULTURE IN THE ANCIENT STATE OF EUROPE, AFTER THE FALL OF THE ROMAN EMPIRE.


[Paragraph 184]
1.F.5.  Some states do not allow disclaimers of certain implied warranties or the exclusion or limitation of certain types of damages. If any disclaimer or limitation set forth in this agreement violates the law of the state applicable to this agreement, the agreement shall be interpreted to make the maximum disclaimer or limitation allowed by the law.
```

Notice that the text is in the original form (no preprocessing or transformation is used in the printout). Carefully follow the steps below to build a similarity model. We have provided some code snippets in the Jupyter notebook and reiterated the expected steps. However, feel free to implement this in whatever steps you prefer.

1. Data Loading and Preprocessing

1. Set random seed with: `import random; random.seed(123)`
2. Open the file with UTF-8 encoding using `codecs`.
3. Partition into paragraphs by splitting at empty lines.
4. Remove paragraphs containing the word "Gutenberg".
5. Tokenize the text into words and remove punctuation and whitespaces.
6. Convert to lowercase and stem words using `PorterStemmer`.
7. Use `FreqDist` from NLTK to compute word frequencies.

2. Dictionary Building

1. Build a dictionary using Gensim: `gensim.corpora.Dictionary`.
2. Filter out stopwords from Textfixer.
3. Convert paragraphs into Bags-of-Words (BOW).

3. Retrieval Models

1. Build a TF-IDF model using `gensim.models.TfidfModel`.
2. Convert BOW to TF-IDF weights.
3. Build LSI model with 100 topics: `gensim.models.LsiModel`.
4. Interpret the first 3 LSI topics.

4. Querying

1. Preprocess the query - "What is the purpose of rent?" into BOW.
2. Convert query to TF-IDF, report weights, and retrieve top 3 paragraphs.
3. Convert query to LSI topics, report weights, and retrieve top 3 paragraphs.
4. Compare LSI results with TF-IDF results.