NLP

Author: Carlos Carret Miranda

University of Havana, Faculty of Mathematics and Computer Science (MATCOM)

1

A simple Python program that computes the diversity_measure is in file diversity_measure.py

2 Questions

Ex 1:

The paper titled "Top2Vec: Learning to Word Embeddings Using Contextual Subgraph Topology" presents an innovative method for learning word embeddings based on the contextual subgraph structure in text networks. Below are the five main steps of the Top2Vec method as described in the paper:

1. **Data Preprocessing**: The first step involves preparing the text data. This includes tokenizing the text, removing stop words, and normalizing words (e.g., converting all words to lowercase). Additionally, nodes and edges in the text network are identified, where nodes represent words and edges represent relationships between words (e.g., co-occurrence in a context).

2. **Create Semantic Embedding**: It requires document and word vectors with certain properties to represent semantic associations, placing semantically similar documents close together and dissimilar ones farther apart in the embedding space. Words should be near documents they best describe. To achieve this, models like doc2vec, specifically the DBOW version, are used, which is similar to the word2vec skip-gram model but swaps the context word for the document vector to predict surrounding words in the context window. Learning involves updating context and document vectors to maximize the probability of the context vector given the surrounding word, resulting in a semantic space where documents are closest to the words that best describe them and far from dissimilar words. It argues that the semantic space generated by word2vec and doc2vec represents topics continuously.

3. **Find Number of Topics**: Discusses the advantages of semantic embedding for learning a continuous representation of topics. In the jointly embedded document and word vector space, documents and words are positioned according to their semantic closeness, with document vectors representing the document's topic. Dense areas of documents in the semantic space indicate highly similar documents sharing a common topic. By calculating the centroid of these vectors, a topic vector is derived, which

is most representative of the dense area. The words closest to this topic vector semantically describe the topic. The assumption behind top2vec is that the number of dense areas equals the number of prominent topics. To identify these dense areas, density-based clustering, specifically HDBSCAN, is used on document vectors. However, the curse of dimensionality poses challenges due to the sparsity of document vectors in high-dimensional spaces. To address this, dimension reduction using UMAP is performed before applying HDBSCAN to find dense clusters of documents.

4. **Calculate Topic Vectors**: focuses on identifying dense clusters of documents and noise documents in the reduced dimension space using HDBSCAN after applying UMAP. These clusters correspond to locations in the original semantic embedding space, effectively labeling each document with either a noise label or a label for its belonging dense cluster. Once labeled, topic vectors can be calculated from the document vectors within each dense cluster. The simplest method involves calculating the centroid of all document vectors in the same dense cluster, though other methods like geometric mean or using probabilities from HDBSCAN's confidence levels could also be viable. Experiments showed that these methods yield very similar topic vectors, suggesting the sparsity of the high-dimensional space might be the reason. Consequently, the centroid calculation method was chosen for its simplicity. The process generates a topic vector for each dense cluster, with the number of dense areas indicating the number of prominent topics in the corpus.

In the semantic space, each point represents a topic, and the word vectors closest to a topic vector are those most semantically representative of it. The distance of each word vector to the topic vector indicates the semantic similarity of the word to the topic. These closest words are considered the most similar to all documents in the dense area, summarizing the common topic. Common words, appearing in most documents, are usually equidistant from all documents in the semantic space, making them unlikely candidates for stop-word removal, a finding confirmed through experiments.

5. **Topic Size and Hierarchical Topic Reduction**: Describes how topic and document vectors enable the calculation of topic sizes and the association of each document with its most semantically similar topic. This process assigns each document to exactly one topic, with the size of each topic determined by the number of documents it encompasses. One benefit of this approach is the ability to hierarchically reduce the number of topics found by top2vec to any desired number less than the initial count. This is achieved by iteratively merging the smallest topic with its most semantically similar topic until the target number of topics is reached. This merging process involves taking a weighted arithmetic mean of the smallest topic's vector and its nearest topic vector, with weights proportional to their respective topic sizes. After each merge, the topic sizes are recalculated. This hierarchical reduction method favors topics

with larger sizes, potentially highlighting the most representative topics of the corpus.

Ex 2:

In the context of the paper "TOP2VEC: DISTRIBUTED REPRESENTATIONS OF TOPICS," diversity refers to the variety and richness of topics within a text corpus that the Top2Vec method aims to capture and represent. The method learns distributed representations of topics, which are essentially vector embeddings that capture the semantic and contextual relationships between different topics. By doing so, Top2Vec can uncover a wide range of topics within a dataset, reflecting the diversity of themes, ideas, and discussions present in the text. This diversity is crucial for understanding the complexity and breadth of the information contained in the text, enabling more nuanced and accurate analysis of the topics discussed.

Ex 3:

The difference between using a frequency file and a language model as input for a topic modeling method lies primarily in the depth and sophistication of the representation of words and their relationships within the text corpus. Both approaches aim to capture the underlying structure and semantics of the text, but they do so in fundamentally different ways.

Frequency File Input

A frequency file typically contains counts of word occurrences within a corpus. This approach treats words as discrete entities without considering their semantic or contextual relationships. The primary focus is on the raw frequency of words, which can be useful for basic statistical analyses or when the goal is to identify the most frequently occurring words or phrases in a corpus. However, this method lacks the ability to capture the nuances of meaning and context that words carry.

Advantages: - Simple and computationally efficient. - Useful for tasks requiring a basic understanding of term prevalence.

Disadvantages: - Ignores semantic and contextual relationships between words. - May not accurately reflect the importance or relevance of words in certain contexts.

Language Model Input

A language model, on the other hand, goes beyond simple frequency counts by representing words in a high-dimensional space where their semantic and contextual relationships are preserved. Language models, such as word embeddings (e.g., Word2Vec, GloVe, FastText), map words to vectors in such a way that semantic similarities between words are reflected in the geometric distances between their corresponding vectors. This allows for a much richer and more nuanced understanding of the text than mere frequency counts.

Advantages: - Captures semantic and contextual relationships between words. - Enables sophisticated analyses that leverage the semantic meanings of words.

Disadvantages: - More computationally intensive to train and use. - Requires a larger amount of text data to generate meaningful embeddings.

Application in Topic Modeling

When applied to topic modeling, a frequency file might be used to identify the most frequent words across documents, which could serve as seed words for discovering topics. However, this approach is limited because it does not account for the semantic connections between words that are crucial for understanding the thematic content of documents.

Conversely, using a language model as input allows topic modeling algorithms to leverage the rich semantic information encoded in word embeddings. This can lead to more coherent and semantically meaningful topics, as the algorithm can better understand the relationships between words and how they contribute to the overall theme of a document or collection of documents.

While both frequency files and language models can be used as inputs for topic modeling, language models offer a significantly deeper and more nuanced understanding of the text, making them preferable for advanced topic modeling tasks.

Ex 4:

Analyzing the results in terms of coherency and diversity for different language models with Top2Vec involves several steps, including running experiments, collecting data, and then analyzing and visualizing the results. Given the constraints of this platform, I'll outline a hypothetical approach to conducting this analysis and interpreting the results, rather than executing the code directly.

Steps for Analysis:

1. **Select Language Models**: Choose four different language models to experiment with. These could vary in complexity, such as simple bag-of-words vs. more sophisticated embeddings like Word2Vec or BERT.
2. **Run Experiments**: For each language model, run Top2Vec with varying numbers of topics (starting from 5 to TMax, where TMax changes depending on the language model used). Collect the coherence scores and diversity scores for each combination.
3. **Collect Data**: Store the results in a structured format suitable for analysis, such as CSV files or a database.

4. **Analyze and Visualize Results**: Use a tool like Matplotlib or Seaborn in Python to plot the coherence and diversity scores against the number of topics for each language model.

Hypothetical Visualization Approach:

Coherence Scores Plot:

For plotting coherence scores, you might create a line plot with the number of topics on the x-axis and the average coherence score on the y-axis. Each line would represent a different language model.

```
import matplotlib.pyplot as plt

# Assuming `results_df` is a DataFrame containing the results
plt.figure(figsize=(12, 6))

for lang_model in ['BagOfWords', 'Word2Vec', 'Glove', 'BERT']:
    model_results = results_df[results_df['LanguageModel'] == lang_model]
    plt.plot(model_results['Number_of_Topics'], model_results['Average_Coherence'], label=la

plt.xlabel('Number of Topics')
plt.ylabel('Average Coherence Score')
plt.title('Coherence Scores by Number of Topics and Language Model')
plt.legend()
plt.show()
```

Diversity Scores Plot:

Similarly, for diversity scores, you could create another line plot with the same axes but showing the average diversity score instead.

```
plt.figure(figsize=(12, 6))

for lang_model in ['BagOfWords', 'Word2Vec', 'Glove', 'BERT']:
    model_results = results_df[results_df['LanguageModel'] == lang_model]
    plt.plot(model_results['Number_of_Topics'], model_results['Average_Diversity'], label=la

plt.xlabel('Number of Topics')
plt.ylabel('Average Diversity Score')
plt.title('Diversity Scores by Number of Topics and Language Model')
plt.legend()
plt.show()
```

Interpreting Results:

- **Best Language Model**: The "best" language model for topic modeling using Top2Vec would likely be the one that achieves the highest balance between coherence and diversity across a range of topic numbers. This could vary depending on the specific goals of the analysis (e.g., prioritizing topic clarity over novelty).

- **Complexity vs. Performance**: While more sophisticated language models (like BERT) might theoretically offer greater precision in capturing semantic relationships, simpler models (like BagOfWords) might perform surprisingly well, especially in contexts where the added complexity does not significantly improve topic modeling outcomes. The optimal choice often depends on the nature of the text corpus and the specific requirements of the analysis.
- **Conclusion**: The best language model for topic modeling using Top2Vec is not necessarily the most elaborate one. The choice should be guided by empirical evidence from your analysis, considering both the coherence and diversity of the topics generated across different models and configurations.