Technical assignment: Semantic search engine

Imane Elbacha

7 November, 2021

1 Introduction: Semantic search engine vs Keyword search engine

Keyword searches are quite literal in the sense that computers find terms wherever they appear—even if part of a larger phrase or used in a different context. This approach is effective if the researcher knows exactly what they are looking for. The problem is words often have multiple meanings, so keyword searches often return irrelevant results (false positives), failing to disambiguate unstructured text.

Keyword searches may also fail to turn up related materials that don't specifically use the search term (false negatives). Under these conditions, researchers can miss pertinent information. There is also the danger of making business decisions based on a less than comprehensive set of search results.

Unlike keyword search, semantic search takes into consideration the researcher's intent to get at the contextual meaning of terms. Semantic search pushes beyond the boundaries of the organization's collective base of understanding to get at information and concepts that haven't been explicitly written into the query. Semantic technology deciphers concepts and meaning by associating search inputs with clarifying terms such as related synonyms that have been built into the system.

This is possible because of the process of semantic enrichment which helps researchers find new ideas and concepts by tagging unstructured text with information about its meaning. To distinguish concepts, semantic technology references vocabularies that contain all known terms for the same thing, relates these entities to each other in hierarchical relationships, and employs algorithms to analyze the context within which those terms appear.

In this technical assignment, we're gonna tackle an approach to build a semantic search engine. The main parts of this assignment are:

1. Data processing
2. Search queries processing
3. algorithm structure and techniques

2 Data processing and cleaning:

Data processing is an important step in text analytics. The purpose is to remove any unwanted words or characters which are written for human readability, but won't contribute to topic modeling. These are the elemnts to take into account in data cleaning

- *Tokenization and De-capitalization:* it's the process of splitting a sentence into a list of words and additionally turning the words into lower case (a step that can be important to remove stopwords) and removing the extra space.
- *Spell and Grammar Correction:* These techniques can be a good way to obtain better results when working with text data, by eliminating the 'noise' in the text
- *Breaking the Attached Words:* for example 'DataScience' it can have a significant impact on the result. The technique is straightforward splitting by the capital letter.(this step can be merged with the first one)

- *Stopwords:* Stopwords are the words which are used very frequently, and they're so frequent, that they somewhat lose their semantic meaning. Words like "of, are, the, it, is" are some examples of stopwords. In this task in particular, it is intersting to remove these elements since they don't have a significant semantic impact. There are two approaches of removing the stopwords. One way is using word frequencies and applying a threshold. The other way is to have a predetermined list of stopwords, which can be removed from the list of tokenized sentences.

- *Lemmatizing/Stemming:* it is the process of linking a list of terms into one root term for example: playing, players ,played have one common root which is play. It is important to take into consideration that there is a risk of loosing some semantic meaning with this process the utility of this step could be determined through tests on data.

For this process there are several libraries in Python that could be useful. For instance, spaCy, CoreNLP, PyNLPI, Polyglot but NLTK and spaCy are the most widely used.

3 Search queries processing:

To give back results that are relevant to the search query, it is important to take the input information through a similar process by : Tokenization and De-capitalization, Spell and Grammar Correction, Breaking the Attached Words, removing Stopwords and Lemmatizing/Stemming if used.

4 Algorithm structure:

Using the data acquired from the pre-procing and cleaning we follow this algorithm to build the semantic search engine:

1. *Design the Vocabulary:* In simple words we build the vocabulary of the given textual data in which all the unique words are given IDs (index in gensim library) and their frequency counts are also stored. For this step we can use gensim library for building the dictionary.

2. *Threshold:* We apply a threshold on word frequencies in the dictionnary by removing words that will not be contributing to the various themes or topics. We take note that the threshold depends on the initial data provided. We remove the words from the data too.

3. *Feature extraction:bag-of-words model* The bag-of-words model is a simplifying representation used in natural language processing and information retrieval (IR). In this model, a text (such as a sentence or a document) is represented as the bag (multiset) of its words, disregarding grammar and even word order but keeping multiplicity. The output is fixed-length vectors with the count how many times each word appears. The doc2bow method, iterates through all the words in the text, if the word already exists in the corpus, it increments the frequency count, other wise it inserts the word into the corpus and sets it frequency count to 1. For example:

- *the cat sat*
- *the cat sat in the hat*
- *the cat with the hat*

Document	the	cat	sat	in	hat	with
the cat sat	1	1	1	0	0	0
the cat sat in the hat	2	1	1	1	1	0
the cat with the hat	2	1	0	0	1	1

- *the cat sat*: `[1, 1, 1, 0, 0, 0]`
- *the cat sat in the hat*: `[2, 1, 1, 1, 1, 0]`
- *the cat with the hat*: `[2, 1, 0, 0, 1, 1]`

4. *Tf-Idf and LSI Model:* Latent semantic indexing (LSI) is an indexing and retrieval method that uses a mathematical technique called singular value decomposition (SVD) to identify patterns in the relationships between the terms and concepts contained in an unstructured collection of text. Plus, tf-idf is a weighting scheme that assigns each term in a document a weight based on its term frequency (tf) and inverse document frequency (idf). The terms with higher weight scores are considered to be more important. At first we apply Tf-Idf to determine the most important words in each document in the data. Once the Tf-Idf is build, pass it to LSI model and specify the number of features to build.

Finally, we will input a search query, process it and the theorized model will return relevant text conversations with "Relevance %" which is the similarity score. The higher the similarity score, the more similar the query to the document at the given index. We can add a function that puts the outputs in order of relevance.