INFORMATION RETRIEVAL (CS F469) Assignment

A News Event Detection Algorithm Based on Key Elements Recognition

Group Members:

2016A7PS0012P Avi Rai

2016A7PS0124P Rahul Khapre

2016A7TS0045P Divesh Uttamchandani

Obtaining Dataset:

The Dataset for this Information Retrieval System is from UC Irvine. The dataset contains links to news articles, we have used requests library for downloading and extraction.

Functionality Implemented:

In this paper, we proposed a network news event detection algorithm combined with Named Entity Recognition(NER). The algorithm clusters network news incrementally, based on single-pass clustering algorithm, inherited the simple principle from it and overcame its Shortcoming.

Data Structures Used:

● Lists in python – List in python were used to manipulate data.

● Dictionary in python - Dictionary in python is equivalent to Hash tables

ALGORITHM USED

INPUT: News Report Document

STEPS:

The flow of our information retrieval system is depicted in brief by the image shown below:

1.Sort all the report documents of the topic in chronological order and process the documents according to the time sequence
2.Process the documents with word segmentation, NER, POS tagging and word frequency statistics, get the characteristic word set and form the content vector and time vector of the document
3.Calculate the similarity between the document and the existing event. If the largest similarity is greater than the threshold θ, join the document into the event set which the largest similarity is corresponded to. After that, update the vector space of the event set
4.If all the similarities are smaller than the threshold θ, create a new event, join the document into the new event and update the vector space of the event set
5.After processing all the report documents, the algorithm ends.

Detailed documentation of the data structures can be found in code in specific methods and classes.

After we have sorted all the report documents in chronological order and process the documents the following steps are involved:

1 Preprocessing

Preprocessing comprises of one or more than one steps of processing on the available data so that it is converted to some usable form. Specifically to our project on Text Classification, a number of preprocessing steps are involved which consists of Extracting text from documents of various formats, Tokenization (N-grams technique), Stopwords Removal, Stemming, Vectorization of features and using TfIdf on Vectors as part of feature transformation.After that we can cluster the documents according to similarity. A detailed analysis of the above steps will follow next.

1.1 Tokenization

Prior to tokenization, text is extracted from documents and stored in files. In lexical analysis, tokenization is the process of breaking a stream of text up into words, phrases, symbols, or other meaningful elements called tokens. The list of tokens becomes input for further

processing such as parsing or text mining. Tokenization is useful both in linguistics (where it is a form of text segmentation), and in computer science, where it forms part of lexical analysis. 1.2 Removal of Stopwords

A stop word is a commonly used word (such as “the”, “a”, “an”, “in”) that a search engine has been programmed to ignore, both when indexing entries for searching and when retrieving them as the result of a search query. We would not want these words taking up space in our database, or taking up valuable processing time. For this, we can remove them easily, by storing a list of words that you consider to be stop words. NLTK(Natural Language Toolkit) in python has a list of stopwords stored in 16 different languages.

2. POS and NER

Named Entity Recognition (NER) refers to recognizing the entities with specific meaning in the text, mainly including names, places, institutions, etc. NER has a wide application in many fields, such as machine translation, quiz systems, information extraction, text parsing and so on.Especially, it plays an important role to promote the practical application of natural language process. In this paper, we adopted NLPIR system in segmentation and NER. This system is released by Institute of Computing Technology,Chinese Academy of Sciences. It works efficiently and effectively in natural language processing.

We use NER and POS tagging on the
segmentation result of news data body text. The tagging content is divided into following categories:

- Words appear in title are tagged as T;
- Named entity including person, location and
- organization are tagged as PER, LOC and ORG
- separately;
- The nouns except named entity are tagged as N;
- All the verbs are tagged as V.
- Then we divide the tagged words into two groups:
- Group A: Title words and named entity.
- Group B: Other nouns and verbs.

We do the word frequency statistics on the two group of words, and sort the words by frequency in a descending order.

3. Feature Extraction - Tf Idf Metric

What is tf-idf frequency: Tf-idf stands for term frequency-inverse document frequency, and the tf-idf weight is a weight often used in information retrieval and text mining. This weight is a statistical measure used to evaluate how important a word is to a document in a collection or corpus. The importance increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus. Variations of the tf-idf weighting scheme are often used by search engines as a central tool in scoring and ranking a document's relevance given a user query.

How to Compute:

Typically, the tf-idf weight is composed by two terms: the first computes the normalized Term Frequency (TF), aka. the number of times a word appears in a document, divided by the total number of words in that document; the second term is the Inverse Document Frequency (IDF), computed as the logarithm of the number of the documents in the corpus divided by the number of documents where the specific term appears.

●

TF: Term Frequency, which measures how frequently a term occurs in a document. Since every document is different in length, it is possible that a term would appear much more times in long documents than shorter ones. Thus, the term frequency is often divided by the document length (aka. the total number of terms in the document) as a way of normalization:

TF(t) = (Number of times term t appears in a document) / (Total number of terms in the document).

IDF: ●

Inverse Document Frequency, which measures how important a term is. While computing TF, all terms are considered equally important. However it is known that certain terms, such as "is", "of", and "that", may appear a lot of times but have little importance. Thus we need to weigh down the frequent terms while scale up the rare ones, by computing the following:

●

IDF(t) = log_e(Total number of documents / Number of documents with term t in it).

Example :- Consider a document containing 100 words wherein the word cat appears 3 times. The term frequency (i.e., tf) for cat is then (3 / 100) = 0.03. Now, assume we have 10 million documents and the word cat appears in one thousand of these. Then, the inverse document frequency (i.e., idf) is calculated as log(10,000,000 / 1,000) = 4.

Thus, the Tf-idf weight is the product of these quantities: tf*idf = 0.03 * 4 = 0.12.

Code Execution

The details on setting up the code and followed by its usage can be found in the README file attached.