CS6370 Natural Language Processing

Assignment 1

Due Date: 20/02/2023 — 11:59 PM

List of Collaborators: ***CS20B070 Sasubilli Yuvan, EE20B029 Chollangi Dheeraj Sai***

---

1. What is the simplest and obvious top-down approach to sentence segmentation for English texts?

   Solution: Sentence segmentation is determining the more extended processing units consisting of one or more words. This task involves identifying sentence boundaries between words in different sentences. Sentences in English languages are delimited by punctuation marks. The most obvious top-down approach to sentence segmentation for English texts is to identify sentence-ender punctuation marks like periods, question marks, and exclamations. So that we can split the text into simpler sentences. Here are a few examples of splitting text into simpler sentences using an above-mentioned approach Text = "Backgammon is one of the oldest known board games. Its history can be traced back nearly five thousand years to archeological discoveries in the Middle East. It is a two-player game where each player has fifteen checkers which move between twenty-four points according to the roll of two dice." Sentences = ["Backgammon is one of the oldest known board games.", "Its history can be traced back nearly five thousand years to archeological discoveries in the Middle East." ," It is a two-player game where each player has fifteen checkers which move between twenty-four points according to the roll of two dice."]

2. Does the top-down approach (your answer to the above question) always do correct sentence segmentation? If Yes, justify. If No, substantiate it with a counter-example.

   Solution: Most Certainly not, Period is not only used for expressing the end of the sentence but also to indicate that some information has been removed, such as letters in a word. In particular, periods are often used in abbreviations, especially Latin abbreviations like, etc., i.e., e.g., c., and et al. Periods are also sometimes used for initials when only the first letter of a word or name is present, such as in H. G. Wells. Three periods together make a new punctuation mark called an ellipsis, which can indicate that some words or entire sentences have been omitted. Similarly in this example "Oh Really! I am surprised with your performance. I wasn't expecting this from you." if we use the above top-down approach we get 3 simpler sentences but the correct split is two simpler sentences.

3. Python NLTK is one of the most commonly used packages for Natural Language Processing. What does the Punkt Sentence Tokenizer in NLTK do differently from the simple top-down approach? You can read about the tokenizer here.

   Solution: Punkt tokenizer utilizes an unsupervised learning algorithm to build a model which is used in dividing the text into a list of sentences. The model has already been trained over a large collection of documents and is capable of handling abbreviation words, collocations, and words that start sentences. A simple top-down approach cannot differentiate between abbreviations and punctuation. Here is a simple example, if we use the punkt sentence tokenizer to split "Mr. Ashbell has been pronounced dead by the doctors. The C.I.D. has taken over the case to find the culprits responsible for the assault.", we will get two sentences. But a simple top-down approach will give us 6 sentences.

4. Perform sentence segmentation on the documents in the Cranfield dataset using: (a) The top-down method stated above (b) The pre-trained Punkt Tokenizer for English State a possible scenario along with an example where: (a) the first method performs better than the second one (if any) (b) the second method performs better than the first one (if any)

Solution: Suppose the discourse contains many abbreviations or instances eg: U.S.A or honorary Mr., Mrs., it can be seen that the naive approach results in many unnecessary splitting of the paragraph. In such cases, the pre-trained punkt tokenizer is the preferred choice. If the paragraph is simple and doesn't involve the usage of periods then it reduces greater amounts of computational complexity but if we consider the accuracy the punkt tokenizer always does the job better than the naive tokenizer.
a) "Mahesh Babu is a professor in the Department of Computer Science at IITM. He is also the author of several books and research papers. He received his doctorate from IITM in 2005." The naive method splits the sentence into three different sentences with low computational complexity. b) "I am a U.S. citizen. I live in Washington D.C. with my family. My favorite city is New York." split using The pre-trained Punkt Tokenizer for English gives ['I am a U.S. citizen.', 'I live in Washington D.C. with my family.', 'My favorite city is New York.'] but the naive method gives wrong splits because of abbreviations.

5. What is the simplest top-down approach to word tokenization for English texts?

Solution: The simplest top-down approach for word tokenization is to use a delimiter to differentiate words and store all the identified words in a list. This can be done by using the split function in python setting whitespace as a delimiter to get the required list of tokens. Example: " The sun rises in the west in mars" is tokenized into ["The", "sun", "rises", "in", "the", "west", "in", "mars"].

6. Study about NLTK's Penn Treebank tokenizer here. What type of knowledge does it use - Top-down or Bottom-up?

Solution: NLTK's Penn Treebank tokenizer utilizes a top-down approach for retrieving tokens from sentences. It makes use of regular expressions to tokenize text as in Penn Treebank, which is a dataset consisting of over four million and eight hundred thousand annotated words maintained by the University of Pennsylvania. It is capable of identifying patterns of punctuations, contractions and other word boundaries to split text into individual tokens. It also applies further rules to convert tokens into their canonical forms, such as "we're" is converted to "we" and "re".

7. Perform word tokenization of the sentence-segmented documents using (a) The simple method stated above (b) Penn Treebank Tokenizer State a possible scenario along with an example where: (a) the first method performs better than the second one (if any) (b) the second method performs better than the first one (if any)

Solution: The nltk tokenizer emulates the tokenizer from the Penn-treebank dataset. The simple approach of using split() does not achieve tokens in a linguistic sense and performs poorly, particularly in more complex or specialized applications where word-splitting accuracy is important.
a)The simple method performs in some cases where A simple example would be the tokenization of "There is a doll, it's very old.".The naive approach will give us "doll," but the nltk tokenizer will give us "doll" and "," for compound sentences connected with ",".
b) The Penn-treebank tokenizer performs better if it's not the above case, A simple example would be "Mr.Mahesh Babu is not the hero for the film Bahubali" where "Mr." is a separate token, which can lead to more accurate and reliable results in text analysis tasks.

8. What is the difference between stemming and lemmatization?

> **Solution:**
> **Stemming :** It usually refers to a crude heuristic process that chops off the ends of words in the hope of achieving this goal correctly most of the time, and often includes the removal of derivational affixes.
> **Lemmatization :** It usually refers to doing things properly with the use of a vocabulary and morphological analysis of words, normally aiming to remove inflectional endings only and to return the base or dictionary form of a word, which is known as the lemma. If confronted with the token saw, stemming might return just s, whereas lemmatization would attempt to return either see or saw depending on whether the use of the token was as a verb or a noun. The two may also differ in that stemming most commonly collapses derivationally related words, whereas lemmatization commonly only collapses the different inflectional forms of a lemma.

9. For the search engine application, which is better? Give a proper justification for your answer. This is a good reference to stemming and lemmatization.

> **Solution:** Stemmers use language-specific rules, but they require less knowledge than a lemmatizer, which needs a complete vocabulary and morphological analysis to correctly lemmatize words. Particular domains may also require special stemming rules. However, the exact stemmed form does not matter, only the equivalence classes it forms. Lemmatizer does full morphological analysis to accurately identify the lemma for each word. Doing full morphological analysis produces at most very modest benefits for retrieval. Either form of normalization tends not to improve English information retrieval performance in aggregate - at least not by very much. Now for the search engine Stemming is preferred over lemmatization as it is more efficient and allows for a wider range of variations to be matched, which is a desired quality in a search engine. Stemming increases recall while harming precision. We know recall measures completeness so stemming would also help the search engine to increase the number of positive predictions.

10. Perform stemming/lemmatization (as per your answer to the previous question) on the word-tokenized text.

11. Remove stopwords from the tokenized documents using a curated list of stopwords (for example, the NLTK stopwords list).

12. In the above question, the list of stopwords denotes top-down knowledge. Can you think of a bottom-up approach for stopword removal?

> **Solution:** We should construct a simple list of prepositions, pronouns, and articles and create a list of common stop words typically removed from text data. We can probably use it to remove stopwords. We should tokenize the text into individual words or tokens using a tokenizer function or a regular expression that splits the text at spaces or punctuation marks. Apply a regular expression to remove all the text's stopwords or non-alphabetic characters. Join the remaining tokens: Once the stop words and non-alphabetic characters have been removed, join the remaining tokens back together into a single string or a list of strings. The bottom-up approach is flexible and customizable whereas The top-down approach is faster and easier to implement

1 References

[1], [2], [3], [4], [5]