CS6370 Assignment 1

Vibhhu Sharma (EE19B128), Purvam Jain (EE20B101)

February 2023

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

Sentence segmentation refers to the task of dividing a block of text into sentences in order to process it further. The simplest and most obvious top-down approach to this in English is to break sentences at "end of sentence punctuation", namely, full stops, question marks and exclamation points.

2 Does the top-down approach (your answer to the above question) always do correct sentence segmentation?

No. This approach would fail to account for the use of abbreviations in text that are also denoted with a ".". For example, the single sentence "Mr. Green is walking in the park." would be segmented into two sentences: ["Mr", "Green is walking in the park].

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?

The Punkt Sentence Tokenizer in NLTK employs an unsupervised algorithm to construct a model for abbreviation words, collocations, and sentence-initial words in order to partition a text into a sequence of sentences. In order to do this, prior to its utilization, it necessitates training on a substantial corpus of plain text in the intended language. As cited in the documentation, unlike our

top-down approach, Punkt knows that the periods in Mr. Singh and F. Scott Fitzgerald do not mark sentence boundaries and that sometimes sentences can start with non-capitalized words as in "i is a good variable name."

4 Perform sentence segmentation on the documents in the Cranfield dataset

4.1 Scenario in which first method outperforms the second

One example where a simple approach involving delimiter detection can perform better than NLTK on the task of sentence tokenization is in the case of text data that does not follow standard grammatical rules, such as social media posts, chat messages, or product reviews.

NLTK uses machine learning models trained on grammatically correct text, and these models may not perform well on data with non-standard grammar or syntax. In contrast, a simple delimiter-based approach can often be more effective for these types of data. Consider the following example:
Had a great time last night!Feeling nostalgic. Gonna have to do this more often.
Our simple top-down approach will give us three sentences: ['Had a great time last night!', 'Feeling nostalgic.', 'Gonna have to do this more often.']
Here, there is no space between the exclamation mark and the beginning of the second sentence and NLTK fails to recognize them as separate sentences, giving output ['Had a great time last night!Feeling nostalgic.', 'Gonna have to do this more often.']

4.2 Scenario in which the second method outperforms the first

This is what we discussed in Question 2.
Example: Tell Mr. Green that I will meet him after 2.5 hours.
Method-1: ['Tell Mr.', 'Green that I will mett him after 2.', '5 hours.']
Method-2: ['Tell Mr. Green that I will meet him after 2.5 hours.']

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

The simplest top-down approach to word tokenization for English texts is to split the text into words based on whitespace and punctuation marks.
A simple algorithm for the same is detailed below:

- Delete leading or trailing whitespace from the text and replace all newline and tab characters with spaces.

- Now, split the text into tokens based on whitespace characters (i.e., spaces). For each token, remove any leading and trailing punctuation marks (e.g., commas, periods, semicolons, etc.).

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

NLTK's Penn Treebank tokenizer uses a top-down approach.It uses an existing set of rules to determine the definition of a word rather than learning from data. This tokenizer performs the following steps:

- split standard contractions, e.g. don't -¿ do n't and they'll -¿ they 'll
- treat most punctuation characters as separate tokens
- split off commas and single quotes, when followed by whitespace
- separate periods that appear at the end of line

7 Perform word tokenization of the sentence-segmented documents

7.1 Naive Approach versus NLTK Tokenizer

Example: This bread, costs $2.01.
Method-1:["This","bread","costs","201"]
Method-2: ["This","bread",",","costs","$","2.01","."]
From the above example we can clearly see that disregarding all punctuations, as the first approach suggests, leads to erroneous tokens when dealing with numbers or currency. While the second approach unnecessarily accounts for all punctuations, which are not as important. Also, the first approach can give wrong outputs in cases as: *'s* is used in ["Mr.Greene's son", "Mr.Greene's going to market."]. Here the first approach will disregard *'s* and concatenate it with the base word while second approach considers it to be a separate token which preserved the meaning.

8 What is the difference between stemming and lemmatization?

Stemming and lemmatization are two techniques used to reduce words to their base or root form. Stemming chops off the suffix from a word to get to its base form or stem by applying a set of rules.Stemming is a quick and computationally efficient method, but it can sometimes produce words that are not

actually words or are not the base form of the original word. Lemmatization is a more complex process that involves identifying the base form of a word based on its context in the sentence and its part of speech. This involves using a vocabulary and morphological analysis to determine the lemma or base form of the word.Lemmatization produces more accurate results than stemming but is slower and more computationally intensive.For example: stemming "mice" might give us "mic" while lemmatization gives "mice" or if we stem the word "went" we might get "we/wen" while lemmatization outputs "go".

9 For the search engine application, which is better? Give a proper justification to your answer.

While stemming is computationally efficient and produces fast results, it prioritizes recall over precision and often can produce unrelated results due to mismatched meanings or generation of non-existent words. The Cranefield Dataset consists of a corpus of scietific papers on aerodynamics; lemmatazitaion should be used. Lemmatization is more effective as it can be customized for domain specific vocabulary to produce more coherent and precise outputs. The Cranefield Dataset also comprises of technical jargon and complex vocabulary related to aerodynamics, which makes lemmatization a better method to reduce the variability of words in such a vast corpus. For example, words as: ["airplane","plane","aircraft","aeroplane"] can all be reduced to same base form using lemmatization while stemming fails to do so.

10 Can you think of a bottom-up approach for stopword removal?

Stopwords are words that are commonly used in natural language but are usually considered to be semantically meaningless. Since these words are frequently used in a corpus we can come up with a frequency based approach to detect and remove such tokens.
Let N denote the total number of documents in the dataset and n denote the number of documents in which a word appears. Then we can calculate the inverse document frequency as:

$$idf = log_2(N/n)$$

And then we can remove the words with lower value of idf as that implies they occur in almost every document and provide no semantic benefits.

11 References

- Lecture Notes and Slides

- NLTK Punkt Documentation
- NLTK Treebank Documentation
- Stemming and Lemmatization

Find complete code here : Code Repository