Natural Language Processing (CS6370)
Assignment 1

February 20, 2023

Team Number 42
Pranjul Dubey (ED21S011)
Bhumika Gupta (EE21S060)

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

Ans: The most straightforward top-down approach to perform sentence segmentation for English texts is by splitting the texts using particular punctuation marks such as ".", "!", "?" etc., if they are followed by a capital letter (indicating the start of a new sentence). However, this additional restriction of punctuation marks being followed by a capital letter can be dropped as the lower-casing of the text is one of the most common preprocessing step.

A simple regex for sentence tokenization based on aforementioned punctuation marks is given as follows:

```
Regex = '[.!?]+'
```

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.

Ans: The naive top-down approach mentioned above doesn't always perform correct sentence segmentation. Some of the cases where top-down approach to sentence segmentation fails are:

(a) Extra Spaces: Presence of extra spaces after the ending punctuation marks. This can be addressed by modifying the regex pattern.

E.g., sample sentence from document id 1: " an empirical evaluation of the destalling effects was made for the specific configuration of the experiment . "

(b) Multiple Punctuation Marks: Presence of multiple punctuation marks back to back.

E.g., sample sentence from document id 6: "his solutions were for the three particular cases .. i propose here to give the general solution to this problem, to indicate briefly how it is obtained using the method of reference 2, and to point out that the solutions given by wassermann are incomplete for times longer than the duration of the heat input ."

(c) Abbreviations: Use of punctuation mark(.) for abbreviations.

E.g., query number 60: "is there any simple, but practical, method for numerical integration of the mixing problem (i.e. the blasius problem with three-point boundary conditions) ."

(d) Improper Use of Punctuation marks: False usage of punctuation marks. *E.g.,*

- *query number 52: "what is the available information pertaining to the effect of slight rarefaction on boundary layer flows (the ?slip? effect) ."*
- *query number 170: "why do users of orthodox pitot-static tubes often find that the calibrations appear to be,. - (a) significantly different from those formerly specified, (b) wildly variable at low reynolds numbers ."*

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.

Ans: The Punkt Sentence Tokenizer in NLTK performs the sentence segmentation using a language model trained on a huge collection of plain text using unsupervised machine learning to learn the abbreviation words, collocations, and words that start sentences.

This tokenizer has been designed to work better than the top-down approach as it uses the information gathered from the corpus for the task. It mitigates the problems (above) to some extent.

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)

Ans: The instances where

(a) top-down approach works better than the punkt tokenizer:

A sample sentence from document id 9:

" it was found that the boundary layer was laminar at reynolds numbers of at least 5 x 10 . transverse contamination caused by the turbulent boundary layer on the tunnel sidewall originated far downstream of the flat plate leading edge at reynolds numbers of 1.5 to 2 x 10, and spread at a uniform angle of 5 compared to 9 degree in low-speed flow . "

- Top-down approach tokenized sentences:
  1. ' it was found that the boundary layer was laminar at reynolds numbers of at least 5 x 10 '
  2. ' transverse contamination caused by the turbulent boundary layer on the tunnel sidewall originated far downstream of the flat plate leading edge at reynolds numbers of 1'
  3. '5 to 2 x 10, and spread at a uniform angle of 5 compared to 9 degree in low-speed flow '
- NLTK Punkt tokenized sentences:
  1. 'it was found that the boundary layer was laminar at reynolds numbers of at least 5 x 10 . transverse contamination caused by the turbulent boundary layer on the tunnel sidewall originated far downstream of the flat plate leading edge at reynolds numbers of 1.5 to 2 x 10, and spread at a uniform angle of 5 compared to 9 degree in low-speed flow .'

As seen above, The first sentence has been tokenized correctly by top-down approach by splitting the text after full stop (highlighted). However, Punkt tokenizer ignores this.

(b) punkt tokenizer works better than the top-down approach:

A sample sentence from document id 6:

"his solutions were for the three particular cases .. i propose here to give the general solution to this problem, to indicate briefly how it is obtained using the method of reference 2, and to point out that the solutions given by wassermann are incomplete for times longer than the duration of the heat input ."

- Top-down approach tokenized sentences:
  1. ' his solutions were for the three particular cases'
  2. ' i propose here to give the general solution to this problem, to indicate briefly how it is obtained using the method of reference 2, and to point out that the solutions given by wassermann are incomplete for times longer than the duration of the heat input '

- NLTK Punkt tokenized sentences:
    1. 'his solutions were for the three particular cases.. i propose here to give the general solution to this problem, to indicate briefly how it is obtained using the method of reference 2, and to point out that the solutions given by wassermann are incomplete for times longer than the duration of the heat input .'

As seen above, The first sentence has been tokenized wrongly by top-down approach by splitting the text after encountering full stop. However, the punkt tokenizer correctly recognizes that it is not the end of sentence and hence doesn't segment the sentence.

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

Ans: The simplest and most obvious top-down approach to word tokenization is to split the sentences using whitespaces and some common punctuation marks such as comma, semicolon, hyphen, colon, slash, double quotes, single quotes etc., as delimiters. The regex can be written as follows:

```
Regex = '\b\w+\b'
```

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

Ans: NLTK's Penn Treebank tokenizer is based on a top-down approach to word tokenization. It uses a set of regular expressions and heuristics to identify and split words. It recursively splits the sentences into smaller pieces, such as clauses, phrases, and words, following a set of rules are predefined using the grammar/syntax of the language and does not take the actual context/ knowledge from the text itself for tokenization.

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)

Ans: The instances where

(a) top-down approach works better than penn treebank tokenizer:

query number 9:

"papers on internal /slip flow/ heat transfer studies ."

- Top-down approach tokenized words:
  ['papers', 'on', 'internal', 'slip', 'flow', 'heat', 'transfer', 'studies']
- Penn Treebank Tokenizer tokenized words:
  ['papers', 'on', 'internal', '/slip', 'flow/', 'heat', 'transfer', 'studies', '.']

As seen above, the word 'slip' and 'flow' has been correctly tokenized as two words by top-down approach, whereas the Penn Treebank tokenizer tokenizes it wrongly as '/slip' and 'flow/' considering the '/' to be a part of the word.

(b) penn treebank tokenizer works better than top-down approach:

query number 170:

"why do users of orthodox pitot-static tubes often find that the calibrations appear to be,. - (a) significantly different from those formerly specified, (b) wildly variable at low reynolds numbers ."

- Top-down approach tokenized words:
  ['why', 'do', 'users', 'of', 'orthodox', 'pitot', 'static', 'tubes', 'often', 'find', 'that', 'the', 'calibrations', 'appear', 'to', 'be', 'a', 'significantly', 'different', 'from', 'those', 'formerly', 'specified', 'b', 'wildly', 'variable', 'at', 'low', 'reynolds', 'numbers']
- Penn Treebank Tokenizer tokenized words:
  ['why', 'do', 'users', 'of', 'orthodox', 'pitot-static', 'tubes', 'often', 'find', 'that', 'the', 'calibrations', 'appear', 'to', 'be', ',', '.', '-', '(', 'a', ')', 'significantly', 'different', 'from', 'those', 'formerly', 'specified', ',', '(', 'b', ')', 'wildly', 'variable', 'at', 'low', 'reynolds', 'numbers', '.']

As seen above, the word 'pitot-static' has been wrongly tokenized as two words by top-down approach, whereas the Penn Treebank tokenizer tokenizes it correctly as one word.

8. What is the difference between stemming and lemmatization?

Ans: Stemming and lemmatization are techniques used to reduce inflected (or sometimes derived) words to their base form, so that words can be analyzed and compared more easily. The differences between stemming and lemmatization are stated as follows:

Stemming	**Lemmatisation**
Uses crude heuristics	Uses vocabulary and morphological analysis of words
Chops off the suffixes/inflexions, Returns stem of the word	Reduces to base/root word, also called 'lemma'
Removes derivational affixes	Removes inflectional endings only
Reduced word is not necessarily a dictionary word	Reduced word is always a dictionary word
-	Can define POS to obtain proper lemma
Faster implementation	Slower (comparatively)
E.g., cunning $\rightarrow$ *cun*	E.g., cunning $\rightarrow$ *cunning*

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

Ans: A search engine's essential attributes are speed, accuracy, relevance, and scalability. In general, search engines use only a specific language for the query search, like the English language, which is a comparatively simpler language, so some of the times stemming and lemmatization provides exact same results.

Since, in a search engine, both the queries and documents are pre-processed using same methods, so it doesn't make a lot of difference to prefer a specific method to another. Though, in general, Lemmatization is known to give more accurate results than stemming, but it is time taking. Stemming performs good as well, but might sometimes match documents that are irrelevant to the search query. So the actual choice is to be made by using the tradeoff between accuracy and speed.

For the toy search engine (cranfield dataset) with only 1400 documents, it wouldn't take much time for lemmatization as well, but in a real world scenario, it is not realisable. So, we preferred the search performance and scalability, in this assignment and hence used stemming to reduce the word tokens to the root words. However, it might sometimes return less relevant results but it will be put to user to select the best possible one among the selected documents.

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

Ans: Performed using code implementation.

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

Ans: Performed using code implementation.

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

Ans: An intuitive bottom-up approach for stop word removal might be to use TF-IDF values of the word tokens. Words in the corpus which are common and unimportant will be assigned low (less than some threshold) TF-IDF values and hence can be easily dropped.