NLP Assignment 1

Sumanth Doddapaneni - CS21D409
Raghavan AK - CS21S025
Aswanth Kumar M - CS21M010

February 2022

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

The simplest top-down approach is to break sentences at standard English delimiters like ".", "!", "?" etc

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

No. E.g - "Hi, my name is Sumanth, I'm taking the NLP course offered by Prof. Sutanu." - this is broken into 2 sentence ['Hi, my name is Sumanth, I'm taking the NLP course offered by Prof', 'Sutanu']

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 top down approach fails in places where abbreviations are present in text. Punkt Sentence Tokenizer is trained with an unsupervised algorithm that identifies possible abbreviations, collocations in the text.

The naive approach relies on a small set of hard coded delimiters and fails in cases with decimal numbers, abbreviations, etc. Punkt tokenizer is trained on large scale corpora (and can be trained on new languages) and can handle the above mentioned problems.

Punkt Sentence Tokenizer uses a two stage approach to detect sentence boundaries. [1]

- Type based classification stage (Features : Collocational Bond, Length, Internal periods, Occurrences without final period) [5]
- Token based classification stage.

Sentence: "Hi, my name is Sumanth, I'm taking the NLP course offered by Prof. Sutanu."

NLTK tokenizer detects that period in "Prof. Sutanu" do not mark sentence boundary. It tokenizes the sentence into a single sentence.

4 Perform sentence segmentation on the documents in the Cranfield dataset using

4.1 the first method performs better than the second one

- **Output with naive method** - ["what is the available information pertaining to boundary layers on very slender bodies of revolution in continuum flow (the", "transverse curvature effect)"]
- **Output with nltk** - ["what is the available information pertaining to boundary layers on very slender bodies of revolution in continuum flow (the ?transverse curvature effect) ."]

Here due to improper spacing between words and delimiter, nltk segmenter failed to split the sentence at the "?". Here the "?" is probably a typo in the dataset, but the idea still holds, if we don't follow proper spacing between sentence boundaries and the words, punkt fails.

E.g "my name is sumanth.I'm a phd student" - punkt returns this as a single sentence.

4.2 the second method performs better than the first one

- **Output with naive method** - ["effects of leading-edge bluntness on the flutter characteristics of some square-planform double-wedge airfoils at mach numbers less than 15", "4"]

- **Output with nltk** - [”effects of leading-edge bluntness on the flutter characteristics of some square-planform double-wedge airfoils at mach numbers less than 15.4.”]

The decimal number is broken into 2 sentences in naive method, while it’s preserved with nltk tokenizer

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

The simplest top-down approach for word tokenization is splitting at spaces, commas, exclamatory marks(!), slashes(/)

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

Penn Treebank uses a Top Down approach. It uses the semantic and syntactic relations between the words for segmenting them [2].

E.g ”They’ll save and invest more.” is split into [’They’, ”’ll”, ’save’, ’and’, ’invest’, ’more’, ’.’]. It can handle common English contractions

E.g ”hi, my name can’t hello,” is split into [’hi’, ’,’, ’my’, ’name’, ’ca’, ”n’t”, ’hello’, ’,’]. Punctuations are considered as separate tokens

7 Perform word tokenization of the sentence-segmented documents using

7.1 the first method performs better than the second one

- **Output with naive method** - [[”is”, ”there”, ”any”, ”information”, ”on”, ”how”, ”the”, ”addition”, ”of”, ”a”, ””, ”boat-tail”, ””, ”affects”, ”the”, ”normal”, ”force”, ”on”, ”the”, ”body”, ”of”, ”various”, ”angles”, ”of”, ”incidence”, ”.”]]
- **Output with ptb** - [[”is”, ”there”, ”any”, ”information”, ”on”, ”how”, ”the”, ”addition”, ”of”, ”a”, ”/boat-tail/”, ”affects”, ”the”, ”normal”, ”force”, ”on”, ”the”, ”body”, ”of”, ”various”, ”angles”, ”of”, ”incidence”, ”.”]]

Here ptb is not able to detect ”/” as a valid delimiter.

7.2 the second method performs better than the first one

- **Output with naive method** - [[”what”, ”methods”, ”-dash”, ”exact”, ”or”, ”approximate”, ”-dash”, ”are”, ”presently”, ”available”, ”for”, ”predicting”, ”body”, ”pressures”, ”at”, ”angle”, ”of”, ”attack.”]]
- **Output with ptb** - [[”what”, ”methods”, ”-dash”, ”exact”, ”or”, ”approximate”, ”-dash”, ”are”, ”presently”, ”available”, ”for”, ”predicting”, ”body”, ”pressures”, ”at”, ”angle”, ”of”, ”attack”, ”.”]]

Naive method fails when the punctuation is right next to the word without any space

8 What is the difference between stemming and lemmatization?

Both stemming and lemmatization are the process of reducing the words into a base form. Stemming is a heuristic way of this process where it tries to remove derivational affixes. Whereas lemmatization uses vocabulary and morphological knowledge to remove affixes and also convert them to a base form. After the stemming the results words may not be meaningful words but lemmatization will result in a correct dictionary words.
Stemmers (stemming) use language-specific rules, but they require less knowledge than a lemmatizer, which needs a complete vocabulary and morphological analysis to correctly lemmatize words. Stemmers operate on a single word without knowledge of the context, and therefore cannot discriminate between words which have different meanings depending on part of speech. [3, 4]

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

The purpose of both the methods is to reduce words to a base form to index them. The comparison of algorithms depends on the downstream product wthat we are trying to create. For an information retrieval system stemming improves the recall of the result, while lemmatization improves the precision of the result. A stemmer would return “mean” for both meanness and meaning even though they are 2 different words. A lemmatizer would keep these words as themselves. A lemmatizer is less aggressive in terms of conversion to root words, thereby adding to the precision. [4] Given that in a search engine we want higher recall than precision, stemming would be a better choice.

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

The simplest bottom-up approach for stopword removal is to remove most frequent occurring words in the corpus.

A more "invloved" way of doing this is with the help of Inverse Document Frequency (IDF). IDF gives us a measure of whether a term is common or rare in a given document corpus. We can sort the IDF values and remove the most frequently occurring words (words with low IDF).

$IDF(t) = \log(\frac{N}{df+1})$

N is the total number of documents in the corpus, and *df* is the number of documents where the term t appears. $+1$ is added to avoid division with zero in case no document has a particular word.

References

[1] https://www.nltk.org/_modules/nltk/tokenize/punkt.html.

[2] https://www.nltk.org/_modules/nltk/tokenize/treebank.html.

[3] https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html.

[4] https://queryunderstanding.com/stemming-and-lemmatization-6c086742fe45.

[5] Tibor Kiss and Jan Strunk. "Unsupervised Multilingual Sentence Boundary Detection". In: *Computational Linguistics* 32.4 (2006), pp. 485–525. DOI: 10.1162/coli.2006.32.4.485. URL: https://aclanthology.org/J06-4003.