CS5615 - Information Retrieval

(2024)

M.Sc. in Computer Science

Department of Computer Science and Engineering Faculty of Engineering

University of Moratuwa

Sidra Mowlana

1. Tokenizer

A tokenizer is an application that partitions a body of text into elements known as tokens. Tokens are generally word and punctuation and also any other character elements like number. Tokenization is the most fundamental step in many text processing tasks and is particularly important in Natural Language Processing (NLP) because it takes up the text input with smaller portions.

How is it used in the code?

In the code, NLTK's 'word_tokenize()' function is used to tokenize the input text:

```
28    # Tokenization
29    print("Start Tokenizing the data")
30    tokens = word_tokenize(text_data)
```

Here, text_data contains the text from assignment_data.txt, and the word_tokenize() function splits it into individual words.

Discussion:

The tokenizer was effective at splitting formal text like the student feedback and research paper into individual words. However, for the Twitter data, tokenization faced challenges with URLs, hashtags, and abbreviations. For example:

- URLs like 'https://t.co/M5cKGyvV8F' were split into multiple tokens.
- Hashtags like '#immigration' was considered as individual tokens which are either beneficial or not depending on the particular task.
- Abbreviations 'cdnpoli' were tokenized as separate words without much attention being paid to their actual meaning as it shortened forms of a phrase signifying 'Canadian politics'.

2. Stopword Removal

They are termed stopwords that include such words as “the” “is’, “in”, and so on since they do not have a lot of meaning in the processing of texts.

How is it used in the code?

In the code, NLTK’s stopwords list is used to remove these common words:

```
32  # Remove stop words
33  print("Start Removing stop words")
34  stop_words = set(stopwords.words('english'))
35  tokens_without_stopwords = [word for word in tokens if word.lower() not in stop_words]
```

Here the idea, each token is to be compared to the list of stopwords and if a token returns true, then it is to be removed.

Discussion:

Removal of stopwords was useful for all three kinds of texts: Twitter data, feedback and text of the research paper. They found that it is easy to cut down on unnecessary spaces and key them into more important spaces. For instance:

- Twitter Data: Removing stopwords like "is", "and", "the" helped focus on hashtags, mentions, and key topics.
- Student Feedback: It simplified the text by removing unnecessary words, leaving the core feedback intact.
- Research Paper: It helped to focus on technical terms, key concepts, and research ideas by removing words that do not contribute much to the meaning.

3. Spell Correctors

A spell corrector on the other hand is a tool that checks a certain text to determine which of the words has been typed wrongly. Generally, it employs a word list dictionary to check whether a certain word is well structured or not; if it is not well structured then it searches for a probable correction according to the spelling.

In this assignment, pySpellChecker library was applied for spell check which identify misspelt words and suggest corrections. The SpellChecker works after removal of stopwords: it underlines the misspelt words and offers corrections for the typecast word.

How is it used in the code?

In the code, PySpellChecker was used to detect and correct misspelt words:

```
# Spell checking and correcting misspelled words
spell = SpellChecker()
misspelled = spell.unknown(tokens_without_stopwords)
print("Start Correcting")
correct_text = [spell.correction(word) for word in tokens_without_stopwords]
# Filter out None values from the corrected text
correct_text = [word for word in correct_text if word is not None]
```

Here, the SpellChecker identifies misspelt words from the list tokens_without_stopwords (i.e., words after stopword removal) and suggests corrections.

Discussion:

Spell correction especially benefited formal text like the research paper and the feedback given to students where possible spelling mistakes were likely to be typos. On one side, it was rather successful when processing keywords stemming from blogs and articles; on the other side, it failed when it came to Twitter data because of excessive use of slang/abbreviations, hashtags not found in dictionaries. For instance:

- The attempt to correct more formal words that should be written in English, such as “undersatand”, to “understand” was effective.
- It was particularly problematic with Twitter-specific terms such as full form of ‘cdnpoli’ or ‘@Shawhelp’ because these are personal names of the people or popular abbreviations and are not easily definable according to normal corpus.

4. Stemmer

The process of cutting text down to base or root form is referred to as stemming. A stemmer strips off the prefixes or suffixes to return any form of a word to its stem form. For instance, the word can be in the context with “running”, “ran”, or “runner”, but will be reduced to “run” of the domain.

How is it used in the code?

In the code, the PorterStemmer from NLTK is used to perform stemming:

```
45    # Stemming the corrected words
46    stemmer = PorterStemmer()
47    stemmed_words = [stemmer.stem(word) for word in correct_text]
```

Here, each word from correct_text (the corrected text after spell-checking) is passed through the stemmer, which removes suffixes and returns the base form of the word.

Discussion:

Stemming worked correctly because it brought all words to its base form; however, it is a radical technique that leads to non-string formation. For example:

- Formal Text: Stemming was fairly successful for words such as "running" that were stemmed and reduced to "run". However, words like organisation might be discarded to the extent of being referred to as ''organ'', which sometimes may hold no meaning at all.
- Twitter Data: Informality of twitter text also impacted stemming in a way since stemming is ineffective for dealing with abbreviations and slangs.
- Research Paper: In the course of the research, stemming has potential problems because truncating words such as 'features' to 'featur' or 'classification' to 'classifi' is not useful.

5. Lemmatizer

Lemmatization is the requirement of words to their base form, also known as lemma depending on the context of usage. Unlike stemming, which we know may produce what are called non-words, lemmatization uses context such as part of speech to ensure that the output of the reduction process is a valid word. For instance, when using a full form of words, the verb of the words will be changed into base form; for instance, instead of using "running", it will be changed to "run", instead of using "better" it will be changed to "good".

How is it used in the code?

In the code, Spacy's Lemmatizer is used for lemmatization:

```
48    # Lemmatization using Spacy
49    lemmatized_words = [token.lemma_ for token in nlp(" ".join(correct_text))]
```

Here, the text is passed through Spacy's language model (en_core_web_sm), which identifies the lemma of each word based on its context.

Discussion:

Lemmatization proved to be more effective than stemming, especially for the research paper and student feedback:

- Formal Text: Lemmatization proved itself to be a capable tool at removing inflections, while keeping the meaning of the word intact. For example, “running” is reduced to “run”, while “better” to “good”.
- Twitter Data: Informal text was processed more effectively by the action of lemmatization than by stemming, although this task also had problems with abbreviations, hashtags and proper names.
- Research Paper: Applying lemmatization appeared to be highly effective in this sense because it eliminated many of the technical terms while preserving their meaning in a research context.

6. Overall Discussion

6.1. Tokenization Accuracy:

The tokenization was generally good on all the text types though issues arose with the data from Twitter; URL’s, hashtags and abbreviations. Tokenizing more structured text like feedback from students or research papers worked well..

a. Twitter Data

- **Challenges**: Before feature engineering, it is obvious that these are collected raw texts, so Twitter data is rather unrestricted, full of hashtags, mentions, URLs, abbreviations and other typically informal features. Tokenization was good to split words but fails when it comes to URL and Hashtags. For instance, the URL

https://t.co/M5cKGyvV8F. It is always good practice to avoid splitting words into tokens especially which may not be useful.

- **Accuracy**: Tokenization was moderately effective, but the kind of texts found on social media contain noise such as links #hashtags which are not useful for most NLP tasks.

b. Student Course Feedback

- **Challenges**: The feedback data is better structured than the questions with full sentences and small amounts of punctuation. The tokenization process was very effective in most of the cases when splitting the text to the tokens. However, words like
 were not correctly treated and were collapsed into two tokens unnecessarily.
- **Accuracy**: High. Tokenization was good except we had minor issues when handling HTML-like tags.

c. Research Paper

- **Challenges**: Research paper is of formal writing style, organised and follows all the rules of punctuation. The tokenizer was quite precise in this case because such a text has many references to grammatical patterns.
- **Accuracy**: Very high. Tokenization was crisp and clear and we found that research paper text was the least complicated to tokenize.

6.2. Spell Correction Impact:

Spell correction refined the text by previously missed typographical errors such as the feedback and research paper published by the students. But for Twitter data, it has limited effectiveness because it included many non-words and abbreviated words such as slang.

Two types of word correction were performed: One of the methods is the isolated word correction, which operates with each word separately completely neglecting the others, except for the necessity to place it somewhere second, and the other is the context sensitive correction.

a. Isolated Word Correction

- **Twitter Data**: Spell correction was an issue if the text we entered was from the Twitter feed because the text usually contained tweets, slang, abbreviations and hashtag symbols which were not included in the spell checker's database. Some of the words such as "cdnpoli" which means abbreviation for Canadian politics new words were unidentified and either unaltered or misidentified.
- **Student Feedback**: This text was more serious and it had not many abbreviations, that is why isolated word correction fits this text well. Even simple mistakes such as 'undersatand' were changed to 'understand' effectively.
- **Research Paper**: When used, isolated word correction proved most effective since the research paper mode of writing was formal and free of informal abbreviations. Overall the error rate was low and where typographical errors occurred they were eliminated effectively.

b. Context-Sensitive Word Correction

- **Twitter Data:** The idea of context-sensitive correction was beneficial over the simple correction sometimes but the utility failed to address specific acronyms used in Twitter in addition to slangs. or, for example, when the name "cdnpoli" did not have a clear referent.
- **Student Feedback:** This was served by correction with reference to the context such that grammatical errors now did not easily occur. For instance 'lectuers' was changed to 'lectures' depending on the context in which it was used.
- **Research Paper:** In the research paper context sensitive correction was done to ensure that the correct word was selected depending on the context of the sentence. For example, the term "form" did not change its meaning when it was used to refer to "feature" form.

Impact of Spell Corrections:

- **Twitter Data:** Unlike the previous text which was formal, correction of work done in isolation was less efficient as the text was informal. CU helped to some extent with correction as per the context used but found complications as well.
- **Student Feedback:** The results also showed that with regards to both correction methods, the isolated and context-sensitive corrections brought enhancement in the

quality of the generated texts where the context-sensitive correction exhibited better grammatical flow.

- **Research Paper:** Each of these correction types was rather efficient in correcting typographical errors. As for context sensitive correction it slightly increased the accuracy when it comes to choosing the right word to say in the given context.

6.3. Stemming and Lemmatization Suitability:

Stemming is best for simple word-processing-oriented requirements such as keyword spotting in which the precise meaning of words is not crucial. However, it can lead to the creation of non-words or very different from the originals words, which is convenient in simple or informal text, but can become a difficulty in academic or texts with the use of many already inflected words.

- **Twitter Data:** It was discovered that stemming was too aggressive for the Twitter data. This was fine for transforming words like “running” to “run” but was not good with handling abbreviations and hashtags. For instance, ‘cdnpoli’ was changed to ‘cdnpoli’; the latter being meaningless.
- **Student Feedback:** For feedback text, stemming worked more or less fine, as shown by its conversion of words such as “understanding,” to “understand”. But it sometimes took words to extremes, thus becoming discrepant to add sense to a statement.
- **Research Paper:** Stemming was possible to use to decide more but coalescing non-words like “classifi” from “classification” reduced the procedure’s standing in academic work.

Lemmatization is much more appropriate for the texts in which the meaning is to be retained, for instance, in case with research papers and structured feedback. It makes sure that words are also dissected as much as possible merely on meaningful units while not being grammatically incorrect.

- **Twitter Data:** Lemmatization was helpful more than stemming for the Twitter information dataset. However, it was still ineffective at handling abbreviations and this operation rid words of suffixes and prefixes preserving their semantics. For

instance, when lemmatization involved “running” the output was commendable in that the result was “run”.

- **Student Feedback:** I found that lemmatization for instance contains words such as “better” which when lemmatized is reduced to “good” which retained its semantic meaning which is important in analysing feedback.
- **Research Paper:** Thus, lemmatization was most useful for the research paper. It maintained the meaning but strip words to their roots. For instance, “features” was transformed to “feature” but in contrast with stemming it did not output non-words making it more appropriate for formal environment.

6.4. Stopword Removal Effectiveness:

Exclusion of stopwords ensured that only important information dominated all the renewed generalised text forms. Especially during the research paper and while marking the students’ feedback, many of the words tend to be real working words, not very significant (i.e., the, is, in).

- **Twitter Data:** In the case of Twitter data, stop word removal could be somewhat useful, but often prepositions and articles eliminated from the text were important for understanding of the message. Thus, such stop words as “the”, “is” have been eliminated successfully, however, since a lot of text contained slang, hashtags, and abbreviations, it became clear that the use of the list of stop words did not sterically affect the content.
- **Student Feedback:** Exclusion of stop words proved satisfactory for the students’ feedback. As the text was highly written, eliminating such stop words as ‘the’ ‘and’ ‘is’ helped make the content of the text more meaningful. Deletion of these words did not influence the main concepts because the whole sentences were provided and within each a clear context was presented.
- **Research Paper:** I observed that stop word removal worked quite well on the text of the research paper. Since the text was more formal, performing stop-word removal did not have a negative impact on the messages’ comprehensibility while reducing their length. The simple preprocessing of eliminating stop words such as ‘the’ or ‘of’ provided aid in concentrating on the substance to be analyzed such as in tasks such as summarization or keyphrase extraction.

7. Conclusion

Every of the pre-processing techniques shown in this assignment has its advantages and is applicable to distinct types of text. Tokenization, spell correction as well as lemmatization are most effective when applied to rather substantial formal text, for instance School papers and students' feedback. Although stemming can be valuable in simple retrieval systems it may cut words too finely eradicating any meaning. Last is the stopword removal that is used to minimise the noise of the text and give importance to the most relevant part of the text.