n-grams

Prof. Eric A. Suess

March 24, 2021

Relationships between words

Moving beyond word counts leads to using *n-grams*.

2-grams or *bigrams* can be used to examine which words tend to follow other words.

bigrams can be used to see negation, for example, the use of the word **not** before another word, such as **happy**. Only using word counts can miss the negation, such as **not happy**, in sentiment analysis.

Tokenizing by n-grams

The *unnets_tokens()* function has an option *token* which change set to *"ngrams"* with $n = 2$ to get *bigrams*.

Counting n-grams

The *separate()* and *unite()* functions can be very useful when working with n-grams.

Now we need to filter out *stop_words* from two sets or words, the first word and the second word in a bigram.

Analyzing bigrams

bigrams can be useful to analyze text data where one of the words in the bigram is the same for the filter value. In the book the authors look at Streets in Jane Austin's books.

To do this we can filter on the second word of the bigram *"street"*.

TF-IDF

The td-idf values can be computed for bigrams also.

bigrams can give context in sentiment analysis

A very useful application of bigrams is to look for negations in text when performing Sentiment Analysis.

Example: The use of the word **not** before another word, such as **happy**. Only using word counts can miss the negation, such as **not happy**, in sentiment analysis.

bigrams can give context in sentiment analysis

bigrams can be used to locate words that should have the **opposite** sentiment score.

In the book the AFINN lexicon is used in the example given, this lexion give numeric sentiment values, with positive and negative numbers.

bigrams can give context in sentiment analysis

The example creates

```
> negation_words <- c("not", "no", "never", "without")
```

and used this like the *stop_words*