NLP Galore

Various Sources

2022-06-05

Contents

Chapter 1

Introduction

This handbook is about natural language processing for internal use only.

Please **do not** cite.

BASICS OF NLP

Chapter 2

Dictionary-Based Methods

Chapter 3

Regression-Based Methods

Chapter 4

Bag of Words: TF-IDF

4.1 Word Tokenization

Tokenizers can easily become complex.

4.1.1 Implementing Tokenization

Several Python libraries implement tokenizers, each with its own advantages and disadvantages:

1. spaCy—Accurate , flexible, fast, Python
2. Stanford CoreNLP—More accurate, less flexible, fast, depends on Java 8
3. NLTK—Standard used by many NLP contests and comparisons, popular, Python

NLTK and Stanford CoreNLP have been around the longest and are the most widely used for comparison of NLP algorithms in academic papers.

4.1.1.1 N-Grams

An n-gram is a sequence containing up to n elements that have been extracted from a sequence of those elements, usually a string.

When a sequence of tokens is vectorized into a bag-of-words vector, it loses a lot of the meaning inherent in the order of those words. By extending your concept of a token to include multiword tokens, n-grams, your NLP pipeline can retain much of the meaning inherent in the order of words in your statements.

4.1.1.2 Techniques for Normalizing the Vocabulary

One can normalize the vocabulary so that tokens that mean similar things are combined into a single, normalized form. Doing so reduces the number of tokens you need to retain in your vocabulary and also improves the association of meaning across those different "spellings" of a token or n-gram in your corpus

Case Folding

Case folding is when you consolidate multiple "spellings" of a word that differ only in their capitalization. We can normalize the capitalization using list comprehension: `[x.lower() for x in tokens]`

Stemming

Another common vocabulary normalization technique is to eliminate the small mean- ing differences of pluralization or possessive endings of words, or even various verb forms. Stemming removes suffixes from words in an attempt to combine words with similar meanings together under their common stem. A stem isn't required to be a properly spelled word, but merely a token, or label, representing several possible spellings of a word.

It's important to note that stemming could greatly reduce the "precision" score for your search engine, because it might return many more irrelevant documents along with the relevant ones.

Two of the most popular stemming algorithms are the Porter and Snowball stemmers. The Porter stemmer is named for the computer scientist Martin Porter. Porter is also responsible for enhancing the Porter stemmer to create the Snowball stemmer.

```
from nltk.stem.porter
import PorterStemmer
stemmer = PorterStemmer()
' '.join([stemmer.stem(w).strip("'") for w in "dish washer's washed dishes".split()]))
```

Lemmatization

If you have access to information about connections between the meanings of various words, you might be able to associate several words together even if their spelling is quite different. This more extensive normalization down to the semantic root of a word—its lemma—is called lemmatization.

Lemmatization is a potentially more accurate way to normalize a word than stemming or case normalization because it takes into account a word's meaning. A lemmatizer uses a knowledge base of word synonyms and word endings to ensure that only words that mean similar things are consolidated into a single token.

So lemmatizers are better than stemmers for most applications. Stemmers are only really used in large-scale information retrieval applications (keyword search).

```
nltk.download('wordnet')
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
lemmatizer.lemmatize("better", pos="a") # if POS is not specified, it assumes noun
```

4.2 Bag of Words

A bag of words is a representation of text that describes the occurrence of words within a document.

We just keep track of word counts and disregard the grammatical details and the word order. It is called a “bag” of words because any information about the order or structure of words in the document is discarded. The model is only concerned with whether known words occur in the document, not where in the document.

4.2.1 N-Gram Adaptation

In general, bigrams make tokens more understandable. Suppose we have two sentences:

- Sentence 1: “This is a good job. I will not miss it for anything”
- Sentence 2: ”This is not good at all”

and let us take the vocabulary of 5 words only: “good”, “job”, “miss”, “not”, and “all.”

In this case, the respective vectors for these sentences are:

- Sentence 1: “This is a good job. I will not miss it for anything”=[**1,1,1,1,0**]
- Sentence 2: ”This is not good at all”=[**1,0,0,1,1**]

Sentence 2 is a negative sentence, yet this distinction is not reflected in the vectors.

Now, suppose instead we use bigrams, in which case Sentence 2 can be broken into: “This is”, “is not”, “not good”, “good at”, and “at all.” In this case, the model can differentiate between sentence 1 and sentence 2.

4.3 TF-IDF

TF-IDF is intended to reflect how relevant a term is in a given document. It is computed by multiplying two different metrics:

- **Term Frequency (TF)** of a word t in document d is given as

$$tf(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}$$

  where $f_{t,d}$ is the raw count of a term in a document. The denominator is the total number of terms in document d.

- **Inverse Document Frequency (IDF)** of a word t in a set of documents D is given as

$$idf(t,D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

  where $N = |D|$ and the denominator is the number of documents where the term t appears.

Then TF-IDF is calculated as:

$$tfidf(t,d,D) = tf(t,d) \times idf(t,D)$$

A high weight in TF-IDF is reached by a high term frequency (in the given document) and a low document frequency of the term in the whole collection of documents

4.4 Limitations of Bag of Words

Although Bag-of-Words is quite efficient and easy to implement, there still are disadvantages.

First, the model ignores the location information of the word. The location information is a piece of very important information in the text. For example “today is off” and “Is today off”, have the exact same vector representation in the BoW model.

Second, the model ignores the semantics of the word. For example, words ‘soccer’ and ‘football’ are often used in the same context. However, the vectors corresponding to these words are quite different in the bag of words model.

Third, the range of vocabulary is a big issue faced by the Bag-of-Words model. If the model comes across a new word, it ends up ignoring the word.

Chapter 5

Basic Word Embeddings: Word2Vec & Glove

Chapter 6

Topic Modelling: LSA & LDA

Chapter 7

Sequence Models: RNNs and LSTMs

Chapter 8

Attention Models and Transformers

Chapter 9

Other Useful Methods for Textual Analysis

9.1 Convolutional Neural Networks (CNNs)

9.2 Hidden Markov Models (HMMs)

APPLICATIONS IN ECON/FINANCE

Chapter 10

Sentiment Analysis

CODE SNIPPETS

Chapter 11

Data Scraping

Chapter 12

Data Cleaning