Bag of Words

December 1, 2020

1 Introduction to NLP

Natural Language Processing (NLP) helps us draw insight from text. In many instances, computers are not able to process and analyze text like humans do. Reading a paragraph can easily make a thesis or main idea clear to a human reader, but a computer may not be as adept. A computer cannot instantly put words and phrases together to extract meaning and sentiment and other things that humans can do with ease. To help a computer learn how to this, linguistics and natural processing come into play.

One of the goals of NLP is to draw meaning and identify sentiment in text. Sentiment analysis refers to the process of interpreting and classifying emotions in subjective data. As opposed to relying on summary features like word count, analyzing the text itself can lead to more accurate conclusions that reveal more about the underlying meaning. The goal of this note is to explore one powerful yet simple way to help us do so: using the Bag of Words model of Featurization. In essence, by exploring which words show up in a text and how frequently these words show up, we can teach a computer to better understand the sentiment underlying a piece of text, for example whether it is negative or positive, as well as approach other NLP problems. After seeing enough training data and their associated sentiment as labelled by humans, a good model can start to predict sentiment from never before seen text.

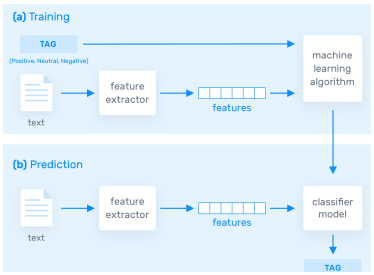

Figure 1: High Level Overview of Sentiment Analysis

2 Bag-of-Words Model

The Bag of Words Model is commonly used in NLP as a featurization to represent text in a document. This featurization can then be passed into any machine learning model, potentially along with other features, such as a logistic regression or support vector machine classifier, to be trained on a dataset and used for predictions.

A document refers to one piece of data (i.e. a singular movie review) and will have associated labels. Before the featurization takes place, the corpus, or collection of all available documents, is examined. The vocabulary, the collection of all distinct words in the corpus, is then developed. The usable vocabulary can be potentially limited to words that we deem important: for example, we can prune all words that show up less than five times in the corpus from our vocabulary. Moreover, further cleaning can be done before establishing this vocabulary such as turning all the words into their lowercase form, spellcheck, and removing punctuation.

Bag of Words takes a document and represents it purely as a collection or 'bag' of words, where we store each word in the document as well as the number of times it occurs. Each document is represented as a vector the size of the usable vocabulary where each index represents a particular word; the number of occurrences for a particular word is stored at its index. Words that are not in the vocabulary are ignored. Only the count of each word is represented; ordering or context of where words occur is ignored in the base featurization. For example, given the following document:

```
one fish two fish red fish blue fish
```

Given a vocabulary of:

```
[the, dog, fish, red, blue, one, two]
```

The bag-of-words featurization would be:

```
[0,0,4,1,1,1,1]
```

Each entry corresponds to one of the words in the usable vocabulary and contains the number of times it appears in the document. In this case, the first non-zero entry corresponds to the word "fish" that appears four times and the last one corresponds to the word "two" which appears one. All zero entries represent words that have not appeared in this specific text.

There are many adjustments to the standard bag-of-words model that can be used to potentially help improve performance in a number of situations.

2.1 Binary Bag-of-Words

One modification of the standard bag-of-words model is binary bag-of-words which simply puts a 1 or 0 for words in the vector instead of keeping track

of the count of words. In many cases, some of which will be explored in the homework, we simply care about whether or not words show up as opposed to how many times they show up. This means that just the presence of the word is enough to contribute to the score and does not need to be scaled by the number of times it appears.

Given the same example as above:

`one fish two fish red fish blue fish`

Given a vocabulary of:

`[the, dog, fish, red, blue, one, two]`

We would now have:

`[0,0,1,1,1,1,1]`

As we see above, words that appear frequently are given the same weight as words that appear a single time in a document. The result is a feature vector that is made up of ones and zeros that define whether or not a word exists in the document. This method may preserve the sentiment of the text when converting it to a feature vector better than standard bag-of-words, because it will not over fit to certain words that appear many times in a single document (i.e. fish in the above case).

2.2 Feature Negation

In addition, we can add more complexity to the standard model by looking at how the negation of words can make features that better understand the sentiment of a text. The goal with this is to make our program understand that words and phrases like "good" and "not good" should be treated as opposites rather than the same word.

In standard bag-of-words, "not good" would map "not" and "good" as separately occurring instances of each word, without taking into account the relationship between the two. In reality, the word "good" is negated, so the sentiment should reflect this negation and be classified appropriately. With added features, we can take into account that "not good" is indeed its own feature that should not add to appearances of "good" in the text, but rather reflect a negative sentiment. One way of doing this is to double the length of our useful vocabulary vector and basically have another word that is "negate_word" We then treat an instance of a word with "not" or "'nt" before it as a "negate_word", and add to the appropriate location in our vector.

2.3 Reducing size of Vocabulary

Since bag-of-words technique looks at every single word, as the document gets larger the bag of words vector can get unnecessarily large. One simple adjustment to help clean the data is to ignore capitalization and discard punctuation.

Since bag of words doesn't care about ordering or context of words, treating "Fish", "fish", and "fish," as the same word can help give a more accurate representation since they all likely mean the same thing.

Another technique that is commonly used is called stemming. Stemming is simply reducing all words to their stem, which can be thought as the basic version of a word. For example, the words "process", "processes", "processed", and "processing" all have the same stem *process*. Given that some words have many different forms, stemming can not only help reduce vocabulary size, but it can also make the model better reflect the meaning of a document, since the counts of the different forms of the word are not spread out.

Finally, short words such as "a", "the", etc., called stop words, often occur the most frequently, which can be a problem since they don't have any meaning. Stop words are usually just filtered out before the text is modeled.

2.4 n-gram

Another variation to the standard bag of words is to use *n-gram*s instead of single words. An *n-gram* is simply a sequence of n words. For example, given the same document from before:

```
One fish Two fish Red fish Blue fish
```

The following would be the *2-grams* used:

```
"One fish", "fish Two", "Two fish", "fish Red", "Red fish", "fish Blue", "Blue fish"
```

Although probably not useful for this example, the idea of *n-grams* is that it better allows for language structure to be represented. All human language have some sort of structure to them, and in many cases ordering and context of words can change the meaning, which is why *n-grams* can be a better model than the standard approach.

3 Other Use Cases

While the name "Bag-of-Words" indicates that this model has a lot to do with text analysis, there are still other use cases for this model as well.

3.1 Computer Vision

Image classification is one of the classic non text use cases for the bag-of-words model. You may hear the model referred to as the bag-of-visual-words in this case. The model works the same way, but with a vocabulary of clustered image features used instead of a vocabulary of words.

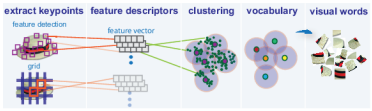

Figure 2: Image Classification Using Bag-of-Visual-Words

Once image features are extracted and clustered to construct a vocabulary, we can use bag of words to featurize each image in a similar way to how we would text data. We can then use labeled data to train a classifier to predict on previously unseen images.

3.2 Image Vocabulary

In a text approach, the unique words act as the vocabulary of the piece of text, which is pretty straightforward. In an image classification approach, the parts of the image make up its vocabulary. As seen in figure 2, an image can be broken down and categorized by its unique parts. For example, a violin has very unique curvature, and holes that work together to define with object. Similarly, a human face has defining characteristics like the positioning of the nose, eyes and mouth. Just like a piece of text is defined by all of its individual words, an image is defined by its defining structure and composition. These image features are then collected for all training data, and a clustering algorithm (k-means, MoG, etc) is used to classify each image feature into a "word". Each cluster then makes up a word that we can use in a bag of words type featurization.

3.3 Feature Extraction

Some of the main ways to choose how to create a vocabulary out of an image are as follows:

- Regular Grid
- Interest Point Detection
- Random Sampling

By using a regular grid, the image is converted into a structured grid, where each feature of the image is one of these rectangular parts of the grid. The vocabulary is the sum of all these parts of the grid.

The interest point detection method can be thought of as splitting up an image in terms of its most stable and formed parts. For example, two different objects in an image may be used as two different points of interest in the image, therefore contributing to different features of the image. In addition, things like corners and textures can be used as points of interest in image because these

are features that are likely to be used to categorize an image.

According to studies, it is shown that by randomly sampling features from an image, one can achieve equal or greater image classification accuracy when compared to a deliberate model like interest point detection or regular grid. Random sampling is often more discriminatory than methods like interest point detection, which leads to better results when classifying images.

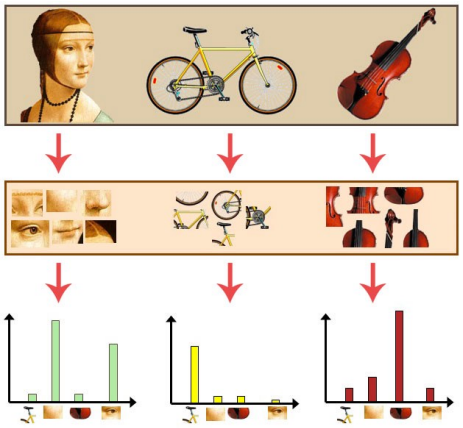

Figure 3: How Images and Features are Extracted and Represented

4 Possible Next Steps

4.1 tf-idf

One problem that can come up in text bag-of-words is that the most frequently occurring words tend to dominate over the other words. To address this problem, a scheme called *tf-idf* can be used, which stands for *term frequency-inverse document frequency*. As the name implies, rather than using the raw count or frequency of a word within a document, tf-idf applies a weight to each word based on the term and document frequency.

Term frequency refers to the frequency of a word within a document. The reasoning behind this weighing scheme is that it gives the most *important* words the highest weight. This scheme assumes that words that have a high term frequency are more *important*, which is reasonable in most cases.

Document frequency refers to the frequency that a word occurs at least among all the documents. By weighing words by the inverse document frequency, it devalues words that appear in many different documents. For example, if analyzing sports news articles, the word *points* likely occurs in most of the articles,

so it's not very useful in distinguishing between two different articles.

For word w in document d, one common weighting scheme is:

$$(1 + \log tf) \cdot \log \frac{1}{df/N}$$

where:
tf is the count of w in d,
df is the number of documents which w occurs in, and
N is the total number of documents we are analyzing.