6COM2000 Advanced Artificial Intelligence

Sentiment Analysis. Naive Bayes for Sentiment Analysis

Olga Tveretina

Overview

Basics of Natural Language Processing (NLP)

Definition

Definition

Natural language processing studies interactions between humans and computers to find ways for computers to process written and spoken words similar to how humans do.

1. An area of AI concerned with the interaction between computers and humans in natural language.

2. The ultimate goal of NLP is to help computers understand language as well as we do. It is the driving force behind things like

   - virtual assistants
   - speech recognition
   - **sentiment analysis**
   - automatic text summarization
   - machine translation

Why NLP Is Difficult?

1. Human language is a complex system, although little children can learn it pretty quickly.
    - there are many different ways to arrange words in a sentence.
    - Words can have several meanings and contextual information is necessary to correctly interpret sentences.
2. We can convey the same meaning in different ways:
    - speech
    - gesture
    - signs

Sentiment Analysis

1. Sentiment analysis is used to identify the affect or emotion (positive, negative, or neutral - subjective information) of the data.
2. For a business, it is a simple way to determine customers' reactions towards the product or service and to quickly pick up on any change of emotion that may require immediate attention.
3. The most basic approach to this problem is to use supervised learning.
4. We can have actual humans to determine and label the sentiment of our data and treat it like a text classification problem. This is exactly what I will go over in this post, and will revisit the topic in the later post to discuss unsupervised methods.

1. Kaggle datasets: https://www.kaggle.com/datasets
   - a wide collection of datasets for various tasks, provided by different companies, organisations, parsed by users from the Internet etc
2. Huggingface datasets: https://huggingface.co/datasets
   - widely used datasets from research papers
3. UCI Machine Learning repository: https://archive.ics.uci.edu/

Sentiment Analysis

Sentiment Analysis

"Sentiment analysis (also known as opinion mining or emotion AI) refers to the use of natural language processing, text analysis, computational linguistics, and biometrics to systematically identify, extract, quantify, and study affective states and subjective information." [Wikipedia]

Example

Given a tweet: "I am happy because I am learning NLP." The goal is to predict whether this tweet is possitive or negative.

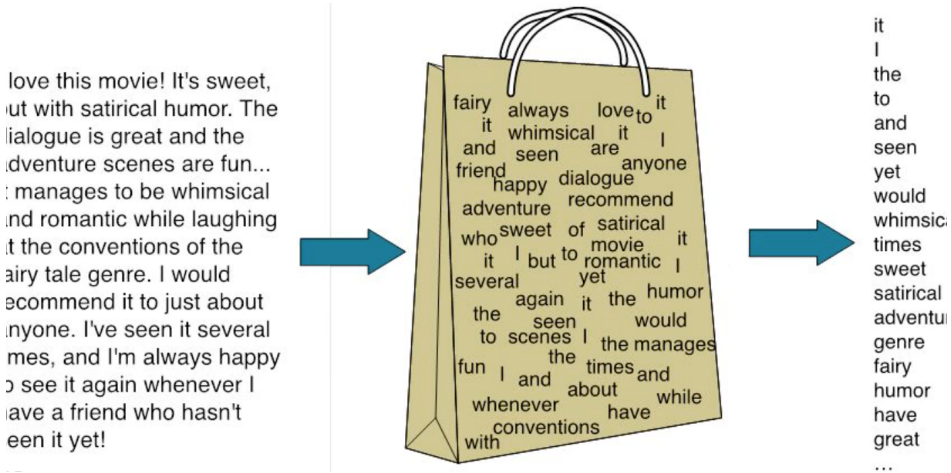
Question: How do we predict whether the tweet is positive or negative?

Approach: Positive or Negative - a binary classification problem.

Vocabulary and Feature Extraction

Feature Extraction - Text Representation - Text Extraction - Text Vectorization

1. What is Feature Extraction from the text?
2. Why do we need it?
3. Why is it difficult?
4. What are the techniques?



Source: https://slideplayer.com/slide/7073400/

Common Terms Used

- **Corpus:** The total number of words present in the whole dataset is known as Corpus.
- **Vocabulary:** Total number of unique words available in the corpus.
- **Document:** There are multiple records in a dataset so a single record or review is referred to as a document.
- **Word:** Words that are used in a document are known as Word.

Bag-of-Words

Representing a text as a vector

1. Bag-of-words (BoW) is a statistical language model used to analyze text and documents based on word count.
2. BoW can be implemented as a Python dictionary with each key set to a word and each value set to the number of times that word appears in a text.

Example

```
{'are':0,
 'many':1,
 'success':2,
 'there':3,
 'to':4,
 'ways':5}

# "many success ways" could be represented
# as the following BoW vector:
[0, 1, 1, 0, 0, 1]
```

Vocabulary and Feature Extraction

Given some text, you can represent it as a vector of dimension $|V|$, where $|V|$ corresponds to the vocabulary size.

Example

For example, you have the following tweet:

"I am successful because I am learning AI"

Then you would need to put a 1 in the corresponding index for any word in the tweet, and a 0 otherwise.

$$\underbrace{[1, 1, 1, 1, 1, 1, 1, 1, 0, 0, \ldots, 0]}_{|V|}$$

As the vocabulary V gets larger, the vector becomes sparse.
It could result in larger training and prediction times.

Negative and Positive Frequencies

The process of converting text data into numbers is called Feature Extraction from the text. It is also called text vectorization.

Negative tweets:

"I am failing because I am not studying AI"
"I am failing"

Positive tweets:

"I am successful because I am studying AI"
"I am successful"

Feature Extraction with Frequencies

Map the word and the positive or negative class it appeared in to the number of times that word appeared in the class.

"I am failing because I am not studying AI"
"I am failing"

Vocabulary	Neg. Frec.
I	3
am	3
failing	2
successful	0
because	1
not	1
studying	1
AI	1

"I am successful, I am studying AI"
"I am successful"

Vocabulary	Pos. Frec.
I	3
am	3
failing	0
successful	2
because	0
not	0
studying	1
AI	1

Feature Vector: $[1, 10, 12]$, where 1 corresponds to bias, 10 - positive feature and 12 is a negative feature.

More about Bias

Feature bias, which reflects measurement errors or biases in human judgement, can negatively impact fairness of machine learning models.

More information can be found at:

`https://towardsdatascience.com/how-to-fix-feature-bias-9e47abccb942`

Text Preprocessing

The first step in any NLP project: putting the data into an analyzable form:

1. Stop word removal: and, is, a, on, etc.
2. Tokenization: a tokenizer breaks unstructured data and natural language text into chunks of information that can be considered as discrete elements. The token occurrences in a document can be used directly as a vector representing that document.
3. Stemming: convert every word to its stem. Like dancer, dancing, danced, becomes 'danc'.
4. Convert all words to lower case

Word Tokenization

1. Dividing a string of written language into its component words.
2. White space is a good approximation of a word divider.

```
from nltk.tokenize import word_tokenize
tokenized_word = word_tokenize(text)
print(tokenized_word)
```

Stop Words Removal

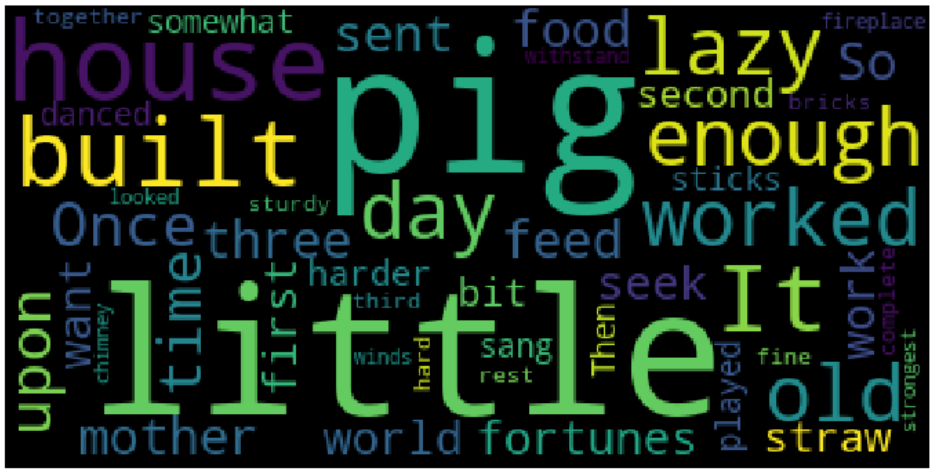
```
from nltk.corpus import stopwords
stopwords = stopwords.words("english")
print(stopwords)
```

```
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

text = "Croatia is my country"

text_tokens = word_tokenize(text)

tokens_without_stopwords = [word for word in text_tokens if
                            not word in stopwords.words
                           ()]

print(tokens_without_stopwords)
```

Data Visualization for Text Data

To visualize text data, generally, we use the word cloud. Words from a given text display on the main chart. Some properties associated with this chart are as follows:

1. More frequent or essential words display in a larger and bolder font.
2. Less frequent or essential words display in smaller or thinner fonts.



Data Visualization for Text Data (cont.)

```
from wordcloud import    WordCloud
import matplotlib.pyplot as plt
wordcloud = WordCloud().generate(text)

plt.figure(figsize = (12, 12))
plt.imshow(wordcloud)

plt.axis("off")
plt.show()
```

Naive Bayes

Naive Bayes Assumptions

1. **The Independence Assumption:**

   The assumption of feature independence in Naive Bayes simplifies the computation and makes the algorithm more efficient.

2. Computational Efficiency:

   By assuming feature independence, Naive Bayes reduces the computational complexity by estimating the individual feature probabilities, which requires less data and simplifies the calculations.

3. Data Sparsity:

   In many real-world applications, datasets may suffer from the problem of data sparsity, where certain combinations of features occur rarely or may not appear in the training data at all.

Naive Bayes Classifier

$P(C_i \mid A)$ represents the probability of C given that $A = a_1, a_2, \ldots a_m$ is observed:

$$P(C_i \mid A) = \frac{P(a_1, a_2, \ldots, a_m \mid C_i) * P(C_i)}{P(A)}$$

Since each "attribute" is conditionally independent, we can directly multiply conditional probabilities of distinct attributes:

$$P(a_1, a_2, \ldots, a_m \mid C_i) = \Pi_{j=1}^{m} P(a_j \mid C_i)$$

That is,

$$P(C_i \mid A) = \frac{\Pi_{j=1}^{m} P(a_j \mid C_i) * P(C_i)}{P(A)}$$

The probability in the denominator part is constant. Hence, we can ignore it when comparing the scores for different classes:

$$P(C_i \mid A) \sim \Pi_{j=1}^{m} P(a_j \mid C_i) * P(C_i)$$

Naive Bayes: Example

Example

Positive tweets:

I am happy because I am learning AI
I am happy, not sad

Negative tweets:

I am sad, I am not learning AI
I am sad, not happy

Naive Bayes: Example (cont.)

Example

We create the following table of the probabilities (computed as frequencies):

Words	**Pos. tw.**	**Neg. tw.**
I	3	3
am	3	3
happy	2	1
because	1	0
learning	1	1
AI	1	1
sad	1	2
not	1	2
	13	**13**

Words	**Pos.**	**Neg.**
I	$\frac{3}{13} \approx 0.24$	$\frac{3}{13} = 0.24$
am	$\frac{3}{13} \approx 0.24$	$\frac{3}{13} = 0.24$
happy	$\frac{2}{13} \approx 0.14$	$\frac{1}{13} \approx 0.08$
because	$\frac{1}{13} \approx 0.08$	$\frac{0}{13} = 0$
learning	$\frac{1}{13} \approx 0.08$	$\frac{1}{13} \approx 0.08$
AI	$\frac{1}{13} \approx 0.08$	$\frac{1}{13} \approx 0.08$
sad	$\frac{1}{13} \approx 0.08$	$\frac{2}{13} \approx 0.14$
not	$\frac{1}{13} \approx 0.08$	$\frac{2}{13} \approx .0.14$
sum =	**1**	**1**

Naive Bayes: Example (cont.)

Example

I am happy, I am learning

$$\frac{0.24}{0.24} \times \frac{0.24}{0.24} \times \frac{0.14}{0.08} \times \frac{0.24}{0.24} \times \frac{0.24}{0.24} \times \frac{0.08}{0.08} \approx 1.75 > 1$$

I am sad ~~today~~

$$\frac{0.24}{0.24} \times \frac{0.24}{0.24} \times \frac{0.08}{0.14} \approx 0.57 < 1$$

Words	Pos	Neg
I	0.24	0.24
am	0.24	0.24
happy	0.14	0.08
because	0.08	0
learning	0.08	0.08
AI	0.08	0.08
sad	0.08	0.14
not	0.08	0.14

Naive Bayes's Inference

For inference, you need to compute the following:

$$\frac{P(pos)}{P(neg)} \prod_{i=1}^{n} \frac{P(w \mid pos)}{P(w \mid neg)}$$

For balanced datasets, $\frac{P(pos)}{P(neg)} \approx 1$. That is, it is sufficient to compute the following:

$$\prod_{i=1}^{n} \frac{P(w \mid pos)}{P(w \mid neg)}$$

How to Improve Naive Bayes Classification Performance?

Zero Probability Problem

In a bag of words model, we count the occurrence of words. If the occurrences of word w in training are 0 then $P(w|pos) = 0$ and $P(w|neg) = 0$.

1. What if a word in a review was not present in the training dataset?
2. **The problem of zero probability. So, how to deal with this problem?**
3. Laplacian smoothing is a technique that prevents zero probability estimates.

Words	Pos	Neg
I	0.24	0.24
am	0.24	0.24
happy	0.14	0.08
because	0.08	**0**
learning	0.08	0.08
AI	0.08	0.08
sad	0.08	0.14
not	0.08	0.14

Using Laplace smoothing, we can represent

Laplacian Smoothing

We compute the conditional probability of a word given a class as follows:

$$P(word \mid class) = \frac{\text{Number of occurences of } w \text{ in the class}}{\text{Class size}}$$

1. If a word does not appear in the training vocabulary, then it gets a probability of 0.
2. The concept is to add a small positive value to each of the existing conditional probability values to avoid zero values in the probability model.

We can add smoothing as follows:

$$P_{smoothing}(word \mid class) = \frac{\text{Number of occurences of } w \text{ in the class} + \alpha}{\text{Class size} + \alpha V}$$

where V is the number of unique words in the vocabulary.

Laplacian Smoothing (cont.)

1. We can add smoothing as follows:

$$P_{smoothing}(word \mid class) = \frac{\text{Number of occurences of } w \text{ in the class} + 1}{\text{Class size} + V}$$

where V is the number of unique words in the vocabulary.

2. If we choose a value of α not equal to 0 ($\alpha \neq 0$), the probability will no longer be zero even if a word is not present in the training dataset.

Laplacian Smoothing: Example

Example

The number of unique words in the vocabulary is 8.

Words	Positive class	Negative class
I	$\frac{3+1}{13+8} \approx 0.19$	$\frac{3+1}{13+8} \approx 0.19$
am	$\frac{3+1}{13+8} \approx 0.19$	$\frac{3+1}{13+8} \approx 0.19$
happy	$\frac{2+1}{13+8} \approx 0.14$	$\frac{1+1}{13+8} \approx 0.1$
because	$\frac{1+1}{13+8} \approx 0.1$	$\frac{0+1}{13+8} \approx 0.05$
learning	$\frac{1+1}{13+8} \approx 0.1$	$\frac{1+1}{13+8} \approx 0.1$
AI	$\frac{1+1}{13+8} \approx 0.1$	$\frac{1+1}{13+8} \approx 0.1$
sad	$\frac{1+1}{13+8} \approx 0.1$	$\frac{2+1}{13+8} \approx 0.14$
not	$\frac{1+1}{13+8} \approx 0.1$	$\frac{2+1}{13+8} \approx 0.14$
sum =	**1**	**1**

Laplacian smoothing helps to avoid zero probabilities.

Laplacian Smoothing: Summary

1. Laplacian smoothing is a smoothing technique that helps tackle the problem of zero probability in the Naive Bayes algorithm.
2. Using higher alpha values will push the likelihood towards a value of 0.5 (the probability of a word equal to 0.5 for both the positive and negative reviews).
3. Since we are not getting much information from that, it is not preferable. Therefore, it is preferred to use $\alpha = 1$.

Log-Likelihood

The log-likelihood is the natural logarithm of the likelihood. Why the log is taken?

1. the logarithm transforms a product into a sum
2. the asymptotic properties of sums are easier to analyze
3. products are not numerically stable: they tend to converge quickly to zero or to infinity; sums are instead more stable from a numerical standpoint

Log-Likelihood (cont.)

When n gets larger, we can get numerical flow issues. We introduce the log and obtain the following equation:

$$\frac{P(pos)}{P(neg)} \prod_{i=1}^{n} \frac{P(w \mid pos)}{P(w \mid neg)} = \log \frac{P(pos)}{P(neg)} + \sum_{i=1}^{n} \log \frac{P(w \mid pos)}{P(w \mid neg)}$$

1. $\log \frac{P(pos)}{P(neg)}$ is called the log prior.
2. $\log \prod_{i=1}^{n} \frac{P(w|pos)}{P(w|neg)}$ is called the log likelihood.

Definition

$\log(xy) = \log(x) + \log(y)$

Log-Likelihood (cont.)

Example

$$\ell(w) = \log \frac{P(w \mid pos)}{P(w \mid neg)}$$

$$\sum_{i=1}^{n} \log \frac{P(w_i \mid pos)}{P(w_i \mid neg)} = \sum_{i=1}^{n} \ell(w_i)$$

I am happy, I am learning

$$\sum_{i=1}^{6} \ell(w_i) = 0+0+0.15+0+0+0 = 0.15 > 0$$

Words	Pos.	Neg.	$\ell(w)$
I	0.19	0.19	0
am	0.19	0.19	0
happy	0.14	0.1	0.15
because	0.1	0.1	0
learning	0.1	0.1	0
AI	0.1	0.1	0
sad	0.1	0.14	0
not	0.1	0.14	-0.15

Training

To train your Naiveve Bayes classifier, you have to perform the following steps:

1. Get or annotate a dataset with positive and negative tweets
2. Preprocess the tweets
    1. Lowercase
    2. Remove punctuation, urls, names
    3. Remove stop words
    4. Stemming
    5. Tokenize sentences
3. Compute $P(w \mid pos)$, $P(w \mid neg)$
4. Compute $\ell(w)$

Testing

Example

This example above shows how you can make a prediction given your $\ell(w)$ dictionary.

In this example the log prior is 0 because we have the same amount of positive and negative tweets.

I am happy, I am learning

$$0 + 0 + 0.15 + 0 + 0 + 0 = 0.15 > 0$$

Words	**Pos.**	**Neg.**	$\ell(w)$
I	0.19	0.19	0
am	0.19	0.19	0
happy	0.14	0.1	0.15
because	0.1	0.1	0
learning	0.1	0.1	0
AI	0.1	0.1	0
sad	0.1	0.14	0
not	0.1	0.14	-0.15