Lab 5 report for Machine Learning: "Exracting Features from Text"

Organtzoglou Evangelos Haralambos Malakoudis
Christodoulos Zerdalis

31/10/2024

1 Introduction and Background of the given paper (Chapter 11)

Chapter 11, "Extracting Features from Text Variables," in the "Python Feature Engineering Cookbook" provides practical techniques for transforming text data into structured features for machine learning. The chapter emphasizes methods in Natural Language Processing (NLP) to quantify text complexity and create useful features from unstructured text data.

Key techniques include:

1. **Counting Characters, Words, and Vocabulary:** Basic features derived from text, such as character count, word count, vocabulary size, lexical diversity, and average word length, can indicate the complexity of a text sample.

2. **Estimating Text Complexity:** Sentence count is used to gauge the text's complexity, with methods demonstrated using NLTK for sentence tokenization and pandas for data processing.

3. **Bag-of-Words (BoW) and N-grams:** BoW is a foundational NLP approach capturing word occurrence without considering order. Using n-grams (sequences of n words) further incorporates local word sequence information, enriching the feature space.

4. **Term Frequency-Inverse Document Frequency (TF-IDF):** TF-IDF is introduced to evaluate word importance relative to the entire document collection, mitigating the dominance of common words.

5. **Cleaning and Stemming Text:** Techniques for removing punctuation, converting to lowercase, eliminating stop words, and stemming words are covered. These preprocessing steps enhance the consistency of text for subsequent analysis.

The chapter leverages libraries like `pandas`, `scikit-learn`, and `NLTK` to implement these techniques, transforming text data into numerical features suitable for predictive modeling.

2 Examining every Recipe of the paper separately

In every Recipe we used the same Dataset , mentioned below and the "train" subset of it in particular.

2.1 Recipe-1

In Recipe 1 we were introduced to the fundamentals of Extracting Features from Text. Using pandas of course and also sklearn's "fetch_20newsgroups" Dataset , we saw how to:

- count the number of characters
- count the number of words in the text
- count the number of unique words
- determine the lexical diversity
- determine the average word size

.
After fitting the dataset into a Pandas DataFrame, we implemented various functions in Python to complete the above tasks. In the end, by creating a function called `plot_features`, we visualized the distribution of a specified text feature (`num_words`) across different target categories, using a total of 20 histograms.

2.2 Recipe-2

In Recipe 2 we were introduced to Sentence tokenization. At first we used a simple sample of text that we assigned to a variable named "text". We proceeded by :

- separating the text into sentences
- counting the number of sentences

After that , we went back to our Dataset ("fetch_20newsgroups") , where we first removed the first part of email and then applied the tokenization.

2.3 Recipe-3

In the 3rd Recipe we saw the so called ("CountVectorizer") library of sklearn. What we did next was we removed all punctuation and numbers and then we set up a "bag of words transformer" that:

- Converted all the characters to lowercase
- Removed all "english stopwords" aka all the usually seen english words from our Dataset
- Retains all the words separately that appeared in, at least, 5% of the text pieces

After fitting the Dataframe to our vectorizer , we created a bag of words which in fact is a whole new Dataframe that contains every unique word of our Dataset in the columns and the number of times it appears in the rows.
We did this procedure twice and the second time of creating the Vectorizer we experimented with n_gramms to see how to extract pairs of words.
Then we printed all these unique words too.

2.4 Recipe-4

In Recipe 4 again we started by removing all the punctuation and numbers. In the following code of this recipe we performed the exact same actions as in the 3rd Recipe , although we now used the "TfidfVectorizer" of sklearn.

2.5 Recipe-5

In the last Recipe we focused on using nltk's libraries , those were the stopwords and the SnowballStemmer We started by reviewing two different ways (one new way) of removing punctuation and a new way to remove all numbers. Again we converted all the text to lowercase. We proceeded by removing all the english stopwords , this time by creating a new function exclusively for that. It performed this action by scanning every text word for word and checking if it belonged to the stopwords. After calling the function to apply this check to a small sample , we then tried the whole Dataframe.
Next , we applied stemming to a single word to see how that would work out. After that we had to create another function to do the same to the whole dataset , the same way the last function worked.
Lastly , we printed a sample to see the results of the function.

3 Examining these Recipes on our Dataset

After studying and testing all this code , it was fairly easy to change it so that it fits our Dataset (imdb_dataset.csv).
The main thing we had to do was to include pandas and all the other necessary libraries for each Recipe and obviously fit the Dataset to a pandas Dataframe.

```
import pandas as pd

df = pd.read_csv(our\file_path)
df.rename(columns={'review': 'text'}, inplace=True)
df.head()
```

Using this code what we did was , we renamed the review column to 'text' so that the rest of the code for the Recipes was consistent to our Dataset and to our needs.
We got the expected Results and similarly to the paper code we saw all the different ways of extracting features from text and numerous ways of manipulating it.

References

[1] Soledad Galli. *Python Feature Engineering Cookbook: A complete guide to crafting powerful features for your machine learning models.* 3rd Edition, Packt Publishing, 2023. Foreword by Christoph Molnar, author of *Interpretable Machine Learning* and *Modeling Mindsets.*