What strategies can be used to preprocess the Dark Web texts?

Before we feed the model with our data, we first need to take a few steps to prepare the data. The first thing is to clean or preprocess the data, which we will talk about in this chapter. After that, the text has to be converted into numbers in the form of vectors, so they can be used by algorithms.

The goal of preprocessing is to get documents that are most meaningful to the model we're feeding them to. Therefore, the preprocessing of text for classification can contribute to a higher accuracy score for the model, because we can get a dataset that is more uniform, with each word carrying more meaning.

There are different strategies for preprocessing. The text can be cleaned severely, or left mostly in its original state. The following paragraphs will describe different types of preprocessing that can be applied to the text.

Lower- or uppercase

When working with text in computer language, all characters are transformed into unique codes following the ASCII encoding method. This means that uppercase characters and lowercase characters are encoded differently. When a model is trained using machine learning, it is important for the model to know when two words are the same. Upper- and lowercase characters in a word result in two different words for the algorithm. Therefore, it is important to unify the text. Either in lowercase or in uppercase. In our approach, we use lowercase, because this is easier to read.

Removing unwanted items

When training a model, we want to only feed it features that are important for the classification process. One can argue that some characters, like question marks or semicolons are not relevant in classifying a certain piece of text. However, this is totally dependent on the type of model you are going to train. For example; in the field of chemistry, numbers and horizontal dashes (part of unicode) are definitely saying something about the context. Characters that can be removed are for example; numbers, unicode and punctuation. Furthermore, a predefined list of stop words can be filtered out, or words that are simply not long enough for a specific task.

Stemming and lemmatizing

To make sure all text is as uniform as possible, we can also apply stemming or lemmatization. Both methods attempt to unify words by making them singular or making them a finite verb. Stemming simply replaces or cuts off words. This can give some strange results, but does a good job of unifying text. Lemmatization uses a dictionary to accomplish the same, which is slower, but gives much more realistic words. Again, it is dependent on the model you are trying to train which is best suited to use.

Spelling

In many cases the data will be gathered from sources produced by human input, which can result in spelling errors. To further unify text and remove words that don't mean that much to a model, you can apply a spell check over your data.

Word length

Just like removing certain stop words because they are meaningless, it is possible that a lot of smaller words are less meaningful for the model than larger, more complicated ones. Hereto, one can filter out words with a certain length that is smaller than a certain threshold. Again, this is dependent on the context.

Effectiveness

All these methods can be used in combination with each other to clean the data in different ways. For our project, we created a preprocessing script that allows us to easily preprocess new datasets by easily selecting different approaches.

The different approaches were tested on several algorithms. The outcome however was that the preprocessing, in most cases, didn't really make a difference. The models pretty much had the same accuracy when different cleaning methods were used. For larger datasets, this may make a bigger difference. Even a 1% increase is a welcome one.