NLP Pre-processing

What is text preprocessing, and how does it work?

Simply put, preprocessing text implies transforming it into a form that can be easily analysed. In this context, a task is a concoction of strategy and subject matter. In this case the task is to extract the most important terms from Tweets using the TF-IDF (method).

Preprocessing that is perfect for one purpose may be the bane of another's existence. As a result, keep in mind that text preparation skills cannot be transferred from one project to another.

Take a look at a simple news dataset and see if you can find words that are frequently used. Using stop words in your pre-processing stage means you'll ALREADY be missing out on some of the more common words because you've already removed them from the list. As a result, there isn't a solution that works for everyone.

Techniques for text preprocessing

Preprocessing your text can be done in a variety of ways. Here are a few ideas to get you started.

Lowercasing

Although it's easy to ignore, lowercasing ALL of your text data is a highly effective text preprocessing technique. It's useful for most text mining and NLP tasks, even if your dataset isn't huge. It also helps keep your predicted results consistent.

If the same words in different cases map to distinct lowercase forms, then lowercasing addresses the sparsity problem.

Lowercasing comes in handy while performing a search, as well. Consider searching for documents that contain the word "usa." However, because "usa" was indexed as "USA," there were no hits. Who is to blame now? The person who designed the user interface, or the person who built the search index?

Lowercasing, on the other hand, should be commonplace. Predicting, for example, what programming language a piece of source code will be written in There is a big difference

between the words system in Java and system in Python. When you lowercase the two, they become identical, and as a result, the classifier's predictive capabilities are diminished. Because of this, not all tasks will benefit from lowercasing.

Stemming

Stemming is the process of getting rid of word mannerisms like "troubled" and "troubles" to their root form (e.g. trouble). Here, the "root" isn't necessarily an actual root word, but a canonical variant of the original.

Using a primitive heuristic approach, stemming attempts to convert words into their root form by chopping off their ends. To avoid confusion, the words "trouble", "troubled", and "troubles" might be renamed "troubl instead of trouble" by simply chopping off the ends.

Stemming uses a variety of algorithms. For English, the Porter's Algorithm is the most commonly used because it has been empirically proven to be effective.

The impact of stemming inflected words

Stemming can help with vocabulary standardisation as well as coping with challenges such as scarcity of words. You want to surface documents that mention "deep learning classes" as well as "deep learn classes," even though the latter doesn't seem correct if you search for "deep learning classes." To find the most relevant results, search for all possible spellings of a word.

In contrast to employing better-engineered features and text enrichment approaches like word embeddings, stemming helps improve classification accuracy just minimally.

Lemmatization

Essentially, stemming is the process of removing inflections from a word and mapping it to its root form. Lemmatization, on the other hand, makes an effort to do things correctly. It doesn't just remove parts; it also changes the root of words. For instance, the term "better" would be translated as "good" in this context. For mappings, it may make use of dictionaries like WordNet or some other rule-based techniques. Lemmatization using a WordNet-based technique can be seen in action here.

The Impact of Lemmatization Using WordNet

When it comes to text search and text classification, lemmatization has no real advantages over stemming. To put it another way, depending on the algorithm you use, it could be much slower than using a standard stemmer and you may need to know the word's part of speech to get the

right lemma. Text classification using neural networks is not affected by lemmatization, according to a study.

It's debatable whether or not the higher costs are justified. However, you may always give it a shot and observe how it affects your key performance indicator.

Removal of Stopwords

A language's stop words are a collection of frequently occurring words. Stop words in English include "a," "the," "is," "are," and a variety of others. Stop words are based on the premise that by removing unnecessary words from a text, we can better concentrate on the vital ones.

It's better to surface documents that discuss text preprocessing in a search engine than documents that discuss what is, for example. This can be accomplished by blocking the analysis of all terms in your stop word list. There are several applications for stop words in various fields such as search and text-classification, topic modelling, and topic extraction.

Despite its effectiveness in search and topic extraction systems, stop word removal was found to be a non-issue when it came to classification. The number of characteristics taken into account is reduced, which aids in keeping your models manageable in size.

Stop word lists can be pre-made or you can design one specifically for your website. You can stop word removal in some libraries (such as sklearn) that let you delete terms that appear in a certain percentage of your texts.

Normalization

Text normalisation is a crucial but often-overlooked stage in the preprocessing process. If you want something to be canonical, you have to normalise it. If you want a canonical form of "good," you can use the words "gooood" and "gud." Mapping nearly identical words such as "stopwords" to just a single word, such as "stopwords," is an additional example.

For noisy texts like social media comments, text messages, and blog post comments full of misspellings, acronyms, and out-of-vocabulary terms (oov), text normalisation is critical. This study found that text normalisation improved sentiment categorization accuracy by 4% when applied to Tweets.

Results of Normalizing Text

Take note of how all of the variants map to the same basic structure.

Unstructured clinical texts, where doctors take notes in non-standard methods, can benefit from text normalisation as well. There are times when near synonyms and spelling variances make it useful for subject extraction (e.g. topic modelling, topic modeling, topic-modeling, topic-modelling).

There is no standard mechanism to normalise texts, unlike stemming and lemmatization. Normally, it is determined by the task at hand. Normalizing clinical texts is distinct from normalising SMS text messages.

Spelling-correction-based techniques and dictionary mappings are common approaches to text normalisation. SMT and dictionary mappings are the easiest. This intriguing research examines the normalisation of text messages using a dictionary-based technique versus an SMT approach.

Noise Abatement

When it comes to noise reduction, the goal is to eliminate any letters, numbers, or text fragments that might get in the way of your text analysis. Text preparation is incomplete without noise removal. Also, it's rather domain-specific.

Tweets, for example, could have noise consisting of only special characters excluding hashtags, which denote notions unique to a Tweet. The issue with noise is that it might lead to inconsistent outcomes when applied to subsequent jobs.

Stemming without Noise Removal

Take note of the fact that the above-mentioned raw words all contain some form of background noise. If you try to stem these words, you'll find that the result isn't particularly attractive. The stem of any of them is incorrect. However, after a thorough cleaning, it now appears significantly improved.

Stemming with Noise Removal

One of the first things you should look into when using Text Mining and NLP is noise removal. Getting rid of noise can be accomplished through a variety of means. Additionally, it removes things like punctuation and special characters from the source code while also removing HTML styling and domain-specific keywords (like "RT" for re-post) as well as the source code and headers. It all comes down to what field you're in and what kind of noise your job necessitates.

Text Enrichment / Augmentation

Text enrichment is the process of adding new information to your existing text data. In order to increase your original text's predictive capacity and depth of your data analysis, text enrichment adds additional semantics.

An example of augmentation in information retrieval is broadening a user's search to better match terms. Text mining could be transformed into text document mining analysis if you ask the right questions. While this is illogical to a human, it can assist in retrieving more useful documents.

You have a lot of leeway in terms of how you spice up your writing. Using part-of-speech tagging, you can learn more about your text's words.

A document classification problem might result in a different categorization when the term "book" appears as a noun rather than a verb since one is used in the context of reading while the other is used in the context of reserving things for someone else to read.

Due to the abundance of texts, individuals have begun utilising embeddings to deepen the meaning of words, phrases, and sentences for purposes such as keyword and phrase extraction for search, synthesis, and text generation as a whole. For NLP techniques based on deep learning, a word-level embedding layer is typical. This makes sense. Embeddings can be created and used in downstream operations, or you can start with pre-established embeddings and modify them.

Phrase extraction, expansion with synonyms, and dependency parsing are more methods for enriching your text data. Phrase extraction recognises compound words as a single entity (aka chunking).

Do you require it all?

Yes and no, however if you want good and consistent outcomes, you'll have to put in some effort. I've divided everything down into three categories: Must Do, Should Do, and Task Dependent to give you an idea of where to start. Quantitative or qualitative testing is available for anything task-dependent before deciding whether you need it or not.

In other words, remember that less is more, and make your approach as simple as possible. You will have to peel back more layers if you run across problems when your overhead grows.. Increased overhead

Must Do:

Noise removal
Lowercasing (also can be task dependent perhaps in some cases)

Should Do:

Simple normalization—(e.g. standardize near identical words)

Dependent on the Task:

Advanced normalization (e.g. addressing out-of-vocabulary words)
Stop-word removal
Stemming / lemmatization
Text enrichment / augmentation

As a result, the bare minimum effort required for any operation is to lowercase the text and remove any unnecessary noise. Depending on your industry, noise can mean different things (see section on Noise Removal). Basic normalizing processes for consistency can also be performed, and then more layers can be added in a methodical manner.