TOOL

Choosing a Preprocessing Technique

Rules for Selecting Preprocessing Techniques

These rules are based on a simple foundational principle: “Know your corpus.” Or, as statisticians would say, “First, look at your data.” As computer scientists would say, “GIGO: garbage in, garbage out.”

1. Order matters

For example, some contraction expansion algorithms are implemented for lowercase text and will miss contractions in other lettercasing, such as “I’m” or anything at the start of the sentence. So, check the algorithm at hand and the text to be scrubbed to evaluate which order you should use.

2. Not all preprocessing tasks are needed in every NLP solution

Every preprocessing task costs time, human, and computational resources. For that reason, apply cleaning tools with care and only as needed. Even a brief examination of the corpus can suggest the necessity of some tools and not others.

- For example, in English text, diacritics are unlikely. Thus, introducing an accent removal step would be wasteful.

Every task should yield more information than it loses.

- For example, “An apple farm inspired the company name Apple, Inc.” has “apple” and “Apple” in its vocabulary. If you lowercase the sentence, the vocabulary shrinks, but you can no longer distinguish the two entities, which are very different in meaning. This may or may not be important for your NLP problem. So you should, at least briefly, evaluate the potential gain and loss resulting from this preprocessing step.

3. Pick the best and most efficient preprocessing step

Some preprocessing steps are equivalent or mutually exclusive. Pick the best one if you can afford the computational time and cost in production. This introduces less noise downstream in the work pipeline.

For example, stemming is much coarser than lemmatization in approximating a word's root. However, stemming is fast, works for every word, and does not require parts-of-speech (POS) tagging, a technique you'll explore in the next module. On the other hand, WordNet lemmatization is much slower, requires the WordNet POS tag set, and can only find roots for roughly 150,000 English words in the WordNet database. If engineering and computational time is not of great concern, you could even apply the WordNet Lemmatizer for the relevant words and use stemming for the rest. However, using stemming and lemmatization for the same words is redundant and sometimes even harmful.

The bottom line as you consider these rules is to investigate the corpus. If it is too big, subsample its parts, look at the distribution of common and rare words to better understand the concentration of information in text, forthcoming challenges, and helpful preprocessing techniques. As you apply these methods, check the input and output at every step until you are confident that it works as planned. In this regard, the popular phrase among computer scientists is, "If you have not tested your code, it does not work."

List of Preprocessing Techniques

The preprocessing techniques you have practiced in this course will help you standardize texts and improve your analysis. Reference this list as you preprocess texts in the future:

1. HTML tag removal

This applies to any mixture of structured and unstructured text. Often, a natural text is intertwined with tags and markers that indicate its formatting and relation in some downstream task, such as reader's consumption. The tags can be in the format of HTML, CSS, JSON, XML, LaTeX, programming languages, etc.

Many common blends, such as HTML+text, are supported by packages such as `BeautifulSoup`, but you can use regex rules in any situation.

2. Tokenization

This includes not just word and sentence parsing, but also character parsing, sub-word parsing, phrase parsing (also known as chunking), paragraph parsing, and so on. In more complex documents you might need to parse different parts of documents. For example, patient discharge notes might need to be parsed into the lab result section, doctor's notes section, patient complaints section, etc.

You tokenize a document into fragments that can be evaluated in greater depth in the following steps and result in significant improvement in the final NLP task.

For example, if you want to extract company names from a document, you need to identify entities, which often requires identifying POS tags, which requires word tokenization. This is not the only approach, of course. For example, if you note that every company necessarily has "***Inc***." after it, then you can reliably extract all company names with a simple regex statement.

3. Stopword removal

More generally, this is the removal of any term that does not contribute to our NLP goal. Typically, stopwords include words which are common to every document and do not help distinguish one document or sentence from another. `nltk` has a list of about 120 generic English stopwords, but you should always evaluate whether more words need to be added or some of the words need to be removed from this list.

4. Contraction expansion

Contraction rules are complicated and ambiguous, but their effective application can notably reduce the vocabulary size and improve POS tagging of the identified clitics. A few simple regex rules can be quickly applied, but are not 100% effective. For example, the phrase "we'd" can mean "we had" or "we would." More sophisticated machine learning tools can provide greater accuracy and can even avoid the necessity of such expansion altogether.

5. Spelling correction

This task is most appropriate for data derived from web entry forms, optical character recognition (OCR) feeds, video transcriptions, and other poor quality sources. Additionally, spelling errors are common in blogs, web forums, and content from smaller websites.

High quality and edited texts, such as Wikipedia, books, newspapers, typically have few spelling errors. These documents are either written by professional writers or are continuously edited by the community.

6. POS tagging

POS tagging is an important task required for lemmatization, chunking, parsing, and other tasks. So far, the `SpaCy` package provides a cutting edge out-of-the box tagger, but `nltk` also has some fast and light algorithms. The quality of POS tagging will very much dictate the quality of the tasks that follow.

7. Stemming and lemmatization

Application of either will standardize the morphological word forms, which reduces the quality of POS tagging. However, these methods are very effective in reducing the vocabulary, sometimes by up to 30 to 40%. If POS tagging is needed, consider it before using stemming or lemmatization.

Lemmatization produces the proper root form, but is slower than stemming and often requires POS tags to find the correct root in a database, such as WordNet, which is also limited to about 150,000 English words.

8. Chunking

If nouns represent the document reasonably well, then you can compress the document by extracting "chunk" noun phrases. Similarly, verbs and other types of phrases can be identified. This provides a compressed summary of the text. Also, you might improve document comparison by comparing key phrases, rather than individual words.

9. Dependence and Constituency Parsing.

These two methods can help further decompose a sentence into (semantic, syntactic, morphological, ...) dependency relations or various phrase relations. Extracted components can also be used to summarize the original text for various NLP tasks.