Comparison of WinkNLP and OpenNLP

Natural Language Processing (NLP) is becoming much more prevalent in today's society. Voice to text, digital assistants, even the word processor the author is using to write this paper uses natural language processing to suggest changes to what is written. This paper will compare features, performance, and usage models for two common NLP tools: WinkNLP and OpenNLP. These two tools are very different but serve the same purpose.

WinkNLP is the work of Graype systems. This company is specifically in the quantitative analysis workspace, designing tools specifically for natural language processing and decision making. WinkNLP is written specifically for Javascript (NodeJS). Installation was easy, using NodeJS's npm package manager tool for installation of the primary package, and the require() lines within the Javascript code installs all models and tools required for use.

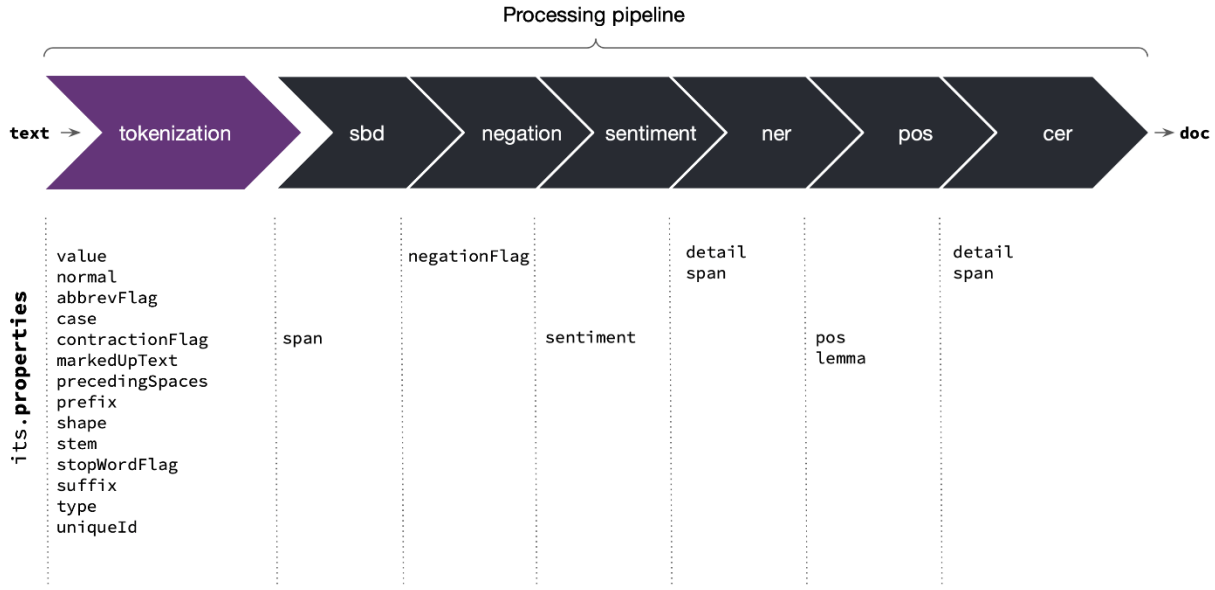
The documentation boasts a top speed of 525,000 tokens per second processing time. The author used the following processing flow to test the speed:

1) Run all constructors/initializers required for the code
2) Load the Cranfield data set (as given in MP2.4)
3) Run the data set through the tokenizer
4) Output the frequency count for each type of word

Following this method, the author found that WinkNLP was able to tokenize 250,405 words in .97 seconds.

Going through the feature set, the author found all of the features he expected to see. WinkNLP has a two-part pipeline. First, the WinkNLP feature "readDoc" processes the document into tokens, with several optional steps:

(Graype Systems, 2021)



Comparison of WinkNLP and OpenNLP

In this picture you can clearly see the processing pipeline. ".readDoc" executes the purple step automatically, with most of the following processing steps being optional.

WinkNLP has the features any data processor typically needs: BM25 vectorizer, parts of speech tags pre-trained for English, negation detection ('not' associated with word), stop word filters, and many more. In addition, WinkNLP leverages two unique helpers: 'its' and 'as.' 'its' allows the filters and tools to access the properties of each token, as listed in the first column of the image above. The 'as' helper specifies how the data from the tool is to be presented/used, such as a frequency table, sets, or counts of unique words.

The learning curve for WinkNLP was not particularly difficult for this novice Javascript programmer. The documentation was fairly clear, several tutorial scripts were available to teach how to use various parts of the tool. The in-line help documentation was clear and easy to use.

OpenNLP solves similar data science problems as WinkNLP. It was written by the Apache corporation, which has written many tools to solve many problems, from document writing, pipelines, and of course natural language processing. OpenNLP is a Java library and standalone CLI tool. This allows developers to either script what they need in whatever language they wish by calling on the CLI tool, or to develop natively in Java, which is a primary language for machine learning development.

Installation of the tool was not as straightforward as WinkNLP. First, the primary package had to be installed. Since this is java, it is available as a .tar.gz file or a .zip compressed file. It is uncompressed in place, and the paths are added to the environment to ease it's use. After the primary tool has been installed, any helper tools you might need, for example a specific language model, is downloaded and put into the directory structure. There is no single package to download and install.

The author wrote a similar script that he used for WinkNLP to test OpenNLP. Using that script, he found that the 250,405 tokens in the Cranfield data set were processed in about .5 seconds, similar to but slightly faster than the WinkNLP tool.

The processing pipeline that OpenNLP uses follows the expected pattern. First, the document is loaded and tokenized. Next, the tokenized array is sent to a number of different tools. These tools can be summarized in the following table:

```
OpenNLP <VERSION>. Usage: opennlp TOOL
where TOOL is one of:
```

Doccat	learnable document categorizer
DoccatTrainer	trainer for the learnable document categorizer
DoccatConverter	converts leipzig data format to native OpenNLP format
DictionaryBuilder	builds a new dictionary
SimpleTokenizer	character class tokenizer
TokenizerME	learnable tokenizer
TokenizerTrainer	trainer for the learnable tokenizer
TokenizerMEEvaluator	evaluator for the learnable tokenizer

Comparison of WinkNLP and OpenNLP

TokenizerCrossValidator	K-fold cross validator for the learnable tokenizer
TokenizerConverter	converts foreign data formats (namefinder con11x pos) to native OpenNLP format
DictionaryDetokenizer	
SentenceDetector	learnable sentence detector
SentenceDetectorTrainer	trainer for the learnable sentence detector
SentenceDetectorEvaluator	evaluator for the learnable sentence detector
SentenceDetectorCrossValidator	K-fold cross validator for the learnable sentence detector
SentenceDetectorConverter	converts foreign data formats (namefinder, con11x, pos) to native OpenNLP format
TokenNameFinder	learnable name finder
TokenNameFinderTrainer	trainer for the learnable name finder
TokenNameFinderEvaluator	Measures the performance of the NameFinder model with the reference data
TokenNameFinderCrossValidator	K-fold cross validator for the learnable Name Finder
TokenNameFinderConverter	converts foreign data formats (bionlp2004, con1103, con1102, ad) to native OpenNLP format
CensusDictionaryCreator	Converts 1990 US Census names into a dictionary
POSTagger	learnable part of speech tagger
POSTaggerTrainer	trains a model for the part-of-speech tagger
POSTaggerEvaluator	Measures the performance of the POS tagger model with the reference data
POSTaggerCrossValidator	K-fold cross validator for the learnable POS tagger
POSTaggerConverter	converts conllx data format to native OpenNLP format
ChunkerME	learnable chunker
ChunkerTrainerME	trainer for the learnable chunker
ChunkerEvaluator	Measures the performance of the Chunker model with the reference data
ChunkerCrossValidator	K-fold cross validator for the chunker
ChunkerConverter	converts ad data format to native OpenNLP format
Parser	performs full syntactic parsing
ParserTrainer	trains the learnable parser
ParserEvaluator	Measures the performance of the Parser model with the reference data
BuildModelUpdater	trains and updates the build model in a parser model
CheckModelUpdater	trains and updates the check model in a parser model
TaggerModelReplacer	replaces the tagger model in a parser model

Comparison of WinkNLP and OpenNLP

All of these tools allow the developer to categorize the generated tokens, as well as filter those tokens given the usual set of categories. In addition, OpenNLP has significantly more trainable routines as compared to WinkNLP.

In comparison, OpenNLP and WinkNLP solves many aspects of the natural language processing tasks. OpenNLP perfoms better on larger documents, according to Pinto, et al. (Pinto, Oliveira, & Alves, 2016) Both were able to access and process the Cranfield test data set without a problem.

In terms of ease of use, WinkNLP had a significantly lower learning curve. The author was able to install and use the tool within 5 minutes of starting. Testing various functions of WinkNLP was easy, and the author was able to extract useful information quickly. OpenNLP was significantly harder to install and use. Several false starts during installation and use had to be recovered. The documentation for installation was not straightforward, and few tutorials were available to guide the author in installation and use. The author has found this to be typical for most Apache products. Given that, OpenNLP seems more useful for large scale development, with a more refined control for machine learning environments.

In terms of accuracy, Pinto, et al, found that OpenNLP had a 99%+-2% precision for large data sets, and a 92%+-2% for twitter-like web data sets. Running WinkNLP through its wink-naive-bayes-text-classifier precision matrix, the author found a 95% precision. While these methods are dissimilar and therefore shouldn't be compared, it can be suggested that WinkNLP and OpenNLP has a similar precision ratio.

Like all tools, the end needs define what tool to use. If a non-dedicated processing pipeline of finite data is needed, the ease of use of WinkNLP should be sufficient for any programmer's needs. It's expandability of tools and models more than justify it's use. However, if unique data sets or criterion, coupled with a dedicated pipeline is needed, OpenNLP's Java processing framework would adequately suit a programmer's needs. OpenNLP has the ability to train for specific data within a dataset that is more powerful than WinkNLP's. This review finds both tools useful for their tasks, and the author will remember both as he progresses in his learning as well as his career.