The Stanford CoreNLP Natural Language Processing Toolkit

<u>Introduction</u>

This toolkit consists of an extensible pipeline that provides core natural language analysis. The toolkit is widely used for production ready systems as well as well as for learning the basics of NLP because of its ease of use and small resource footprint. It was developed in 2006 as a replacement of a glued-together set of multiple NLP components.

Its architecture is simple: Raw text is input into an Annotation object, and then a sequence of annotators add information in an analysis pipeline. The information enriched Annotation object is output to either an XML or a plain text file.

Another advantage is the large coverage of human languages. It started with English, Chinese, German, French and Arabic, and now supports Spanish too.

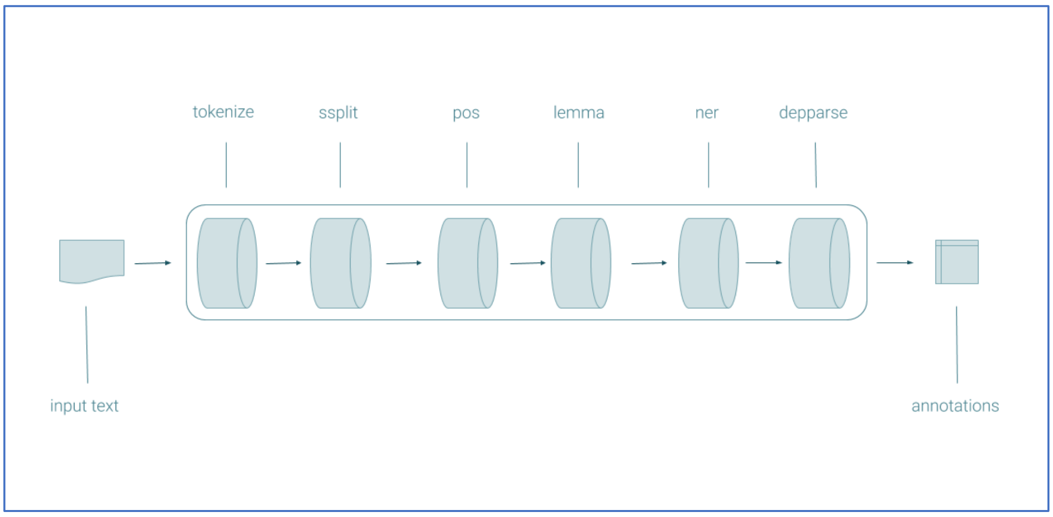
<u>Design and main characteristics</u>

The design motivations were the following:

- To be able to quickly get linguistic annotations for a text
- To provide a common API which isolates the complexity behind
- To have a minimal conceptual footprint (easy to learn)
- To provide a lightweight framework using plain Java objects

The center piece of CoreNLP is the pipeline. The typical execution pipeline follows this sequence:

1. Tokenization
2. Sentence splitting
3. Part of speech tagging (POS)
4. Morphological analysis
5. Named entity recognition
6. Syntactic parsing
7. Coreference resolution
8. Custom annotators like sentiment and gender



The model supports different human languages but comes packaged with models for English. A brief explanation of each of the annotators follows:

tokenize: tokenizes the text in to a series of tokens in Penn treebank style, adapted to handle noisy web text.

cleanxml: removes most of XLM tags from the text

ssplit: splits the tokens into sentences

truecase: determines the likely true case of text written entirely in upper case letters. It uses a Conditional Random Field sequence tagger.

pos: labels all tokens with the part of speech tag, using maximum enthropy.

lemma: generates the lemmas for tokens in the annotation

gender: adds likely gender information to names

ner: recognizes named entities (person, location, organization, etc.) and numerical entities (money, dates, time, etc.)

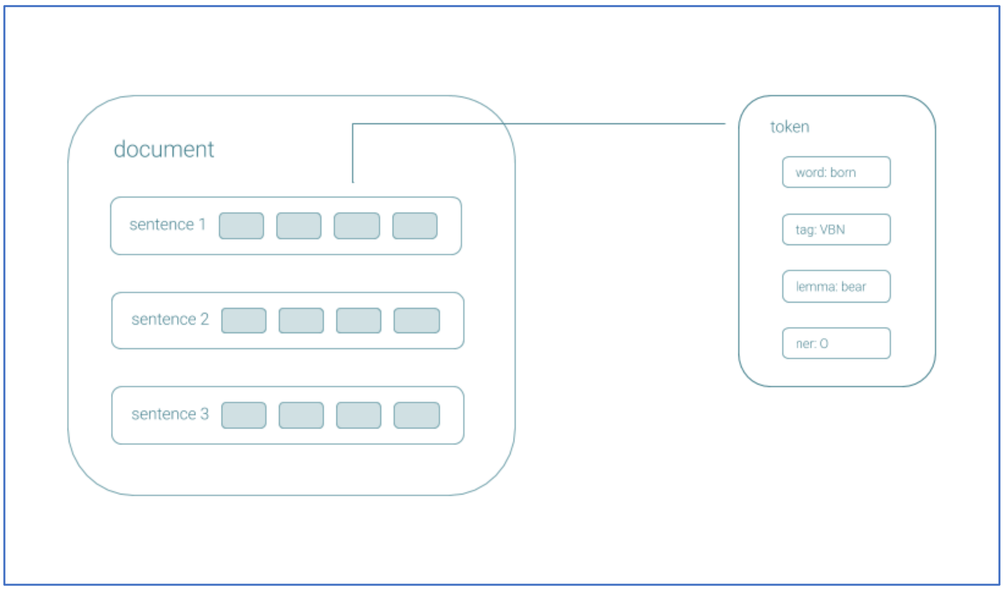
regexner: allows for the inclusion of other types of NER like ideology, religion, nationality, etc.

parse: uses a probabilistic parser to provide full syntactic analysis

sentiment: uses a compositional model (representing an object in terms of its parts and their spatial relation).

dcoref: implements pronominal and nominal coreference resolution

Pipelines produce CoreDocument data objects, which contain all of the annotation information, accessible through a simple API.



Language and operating system support

You need to install Java 1.8+ to use CoreNLP as the code is written entirely in Java. However one can interact with CoreNLP via the its command line or its web service. Many people use it while writing their own code in JavaScript, Python, or other languages. There are wrappers for most data science-oriented languages. The official wrapper for Python is Stanza. CoreNLP works on Linux, MacOS and Windows.

One convenient way of using CoreNLP for Python programmers is to use Stanza, which is a native Python NLP implementation that provides access to all the features of CoreNLP. It starts a CoreNLP server in the background and sends the requests to it and gets back the annotated text without need to deal with any Java code.

Stanza inherits additional functionality from CoreNLP, such as constituency parsing, coreference resolution, and linguistic pattern matching.

Customization

The Core NLP toolkit is easily extensible. While most people will find the provided annotators and their set of optional parameters sufficient for their work, it is easy to add new annotators. An annotator is a class that implements three methods: a method for analysis, a method to define the requirements that the annotation has to satisfy, and a method to determine if those conditions were met.

Conclusion

Stanford CoreNLP is a powerful and very simple to use toolkit with an excellent coverage of human languages. Its framework is small compared to other toolkits, enabling a flat learning curve and facilitating its role as a component of a larger system. It is listed as one of the top 10 toolkits for sentiment analysis in Python. The above mentioned benefits make it a good candidate for our group project: “Sentiment Analysis of Customer Service Tweets” of the JAWs group.

References

1. The Stanford CoreNLP Natural Language Processing Toolkit – by Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, David McClosky – Conference paper Jan 2014
2. CoreNLP home page in Github – https://stanfordnlp.github.io/CoreNLP/index.html
3. Stanza official homepage in Github – https://github.com/stanfordnlp/stanza
4. Stanza Python NLP | Natural Language Processing with Stanza – posted on Medium by Alex Moltzau Jul 15
5. Stanza: A Python Natural Language Processing Toolkit for Many Human Languages – by Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, Christopher D. Manning – Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, July 2020.
6. Overview | Stanza | New Python Natural Language Processing Library | Stanford – posted on Medium by Chen Vu – May 2020