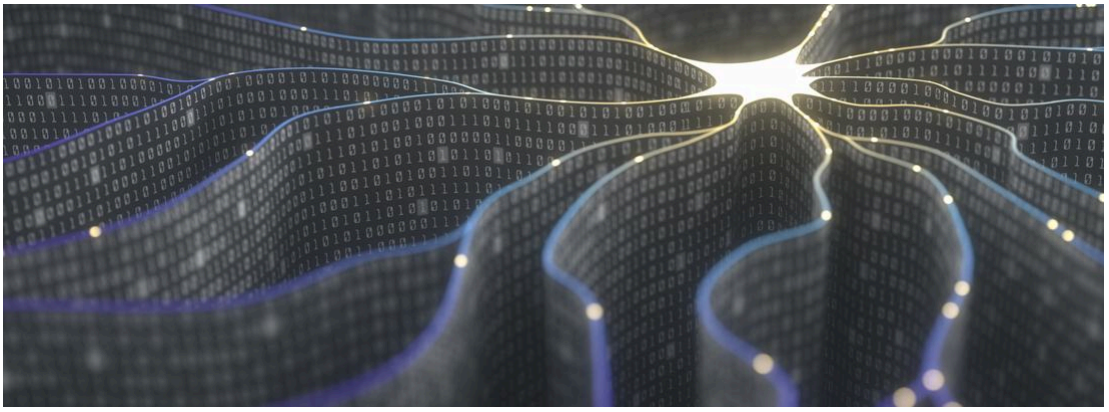
Choosing a Python Library for Sentiment Analysis

Originally published at https://www.iflexion.com/blog/sentiment-analysis-python
Published: April 30, 2019 | By Martin Anderson



Sentiment analysis is one of the hottest topics and research fields in machine learning and natural language processing (NLP). The possibility of understanding the meaning, mood, context and intent of what people write can offer businesses actionable insights into their current and future customers, as well as their competitors.

Constructing an enterprise-focused sentiment analysis system out of the best available frameworks means making some hard choices about the scope, scalability, architecture and ultimate intent of your project.

Because sentiment analysis is still an emerging field, no single solution or approach has won the market yet. The fastest available open-source NLP solution is not the most flexible; the most mature is not the easiest to implement or maintain; some of the most attractive of the other libraries have only a passing disposition toward sentiment analysis.

A better knowledge of the variety of available tools can help you frame the limitations and possibilities for your own future sentiment analysis projects—or at least to inform your strategy when picking partners in ML consulting. So, let's assemble a map of the projects' various capabilities.

A Few Words about Python

The Python programming language has come to dominate machine learning in general, and NLP in particular. Therefore, this article will focus on the strengths and weaknesses of some of the most popular and versatile Python NLP libraries currently available, and their suitability for sentiment analysis.

Due to the open-source nature of Python-based NLP libraries, and their roots in academia, there is a lot of overlap between the five contenders listed here in terms of scope and functionality. Sentiment analysis projects are likely to incorporate several features from one or more of the resources listed here.

1: NLTK (Natural Language Toolkit)

Released:
2001
Created by:
University of Pennsylvania
Highlights:
large numbers of languages and tools supported; well-developed documentation and community

This suite of libraries and applications from the University of Pennsylvania has gained significant traction in Python-based sentiment analysis systems since its conception in 2001. However, its accumulated clutter and educational remit can prove an impediment to enterprise-level development.

The NLTK platform provides accessible interfaces to more than fifty corpora and lexical sources mapped to machine learning algorithms, as well as a robust choice of parsers and utilities.

Besides its provision for sentiment analysis, the NLTK algorithms include named entity recognition, tokenizing, part-of-speech (POS), and topic segmentation. NLTK also boasts a good selection of third-party extensions, as well as the most wide-ranging language support of any of the libraries listed here.

On the other hand, this versatility can also be overwhelming. The sheer variety of some of its tool categories (it has nine stemming libraries as opposed to SpaCy's single stemmer, for instance) can make the framework look like an unfocused grab-bag of NLP archive material from the last fifteen years. This could add a layer of complexity to our project ideation and logistical planning.

The positive side of this is that no competitor to NLTK can boast such a comprehensive and useful base of documentation, as well as secondary literature and online resources. Free ongoing support is provided by a lively Google Group.

Things to Watch Out For

Although NLTK offers Unicode support for multiple languages, setting up non-English workflows is sometimes a more involved process than with other comparable Python libraries. NLTK's out-of-the-box non-English support relies on tertiary mechanisms such as translation layers, language-specific datasets, and models that leverage lexicons or morphemes.

NLTK does not provide neural network models or integrated word vectors, and its string-based processing workflow is arguably behind the times and out of synch with Python's OOP model. NLTK's sentence tokenization is also rudimentary compared to newer competitors.

Final Considerations

If we're training up or onboarding staff that has existing NLTK experience, this very popular set of Python NLP libraries might be the obvious choice; but it comes with a burden of redundancy and complexity that could prove hard to navigate for a new team.

Much of the best of what NLTK has to offer can be accessed in a modular fashion as an external library, as Stanford CoreNLP (see below) has implemented for some of its own components.

Much of the best of what NLTK has to offer can be accessed in a modular fashion as an external library

2: SpaCy

Released:
2015
Created by:
Explosion AI
Highlights:
suitable for industrial solutions; the fastest Python library in the world

With the claim of 'industrial-strength natural language processing', the SpaCy Python library is appealing for sentiment analysis projects that need to remain performant at scale, or which can benefit from a highly object-oriented programming approach.

SpaCy is a multi-platform environment that runs on Cython, a superset of Python that enables the development of fast-executing C-based frameworks for Python. Consequently, SpaCy is the fastest-running solution at the moment according to research by Jinho D. Choi et.al.

Unlike NLTK, SpaCy is focused on industrial usage and maintains a minimal effective toolset, with updates superseding previous versions and tools, in contrast to NLTK. SpaCy's prebuilt models address essential NLP sectors such as named entity recognition, part-of-speech (POS) tagging and classification.

In contrast to its older rival, SpaCy tokenizes parsed text at both the sentence and word levels on an OOP model. It also offers integrated word vectors, Stanford NER and syntactic parsing (including chunking). Enabling sentiment analysis with SpaCy would involve devising your own framework, though; SpaCy, unlike TextBlob (see below), has no native functionality for this purpose.

Things to Watch Out For

However, capable as SpaCy’s models are, we're stuck with their structure. It’s therefore essential to ensure in advance that your long-term goals won’t go out-of-bounds at a later date and become incompatible with this sparse design philosophy.

While SpaCy has an overall speed advantage over its stablemates, its sentence tokenization can run slower than NLTK under certain configurations, which might be a consideration with large-scale pipelines.

Although it demands Unicode input, SpaCy's multi-language support is a work in progress, with models currently available for German, Greek, English, Spanish, French, Italian, Dutch and Portuguese.

With its deliberately lean feature set, SpaCy (as the project website admits) is not an environment suitable for testing different neural network architectures, and is not a good starting point to explore bleeding-edge developments in NLP. SpaCy remains more committed to a consistent platform experience that is focused on the core objectives of its users.

SpaCy is resource-intensive, and requires a 64-bit Python stack as well as higher memory requirements per instance (in the order of 2 or 3 gigabytes) than some of its rivals.

Final Considerations

If your project fits within the deliberate limitations of the SpaCy framework, this may be the most 'production-ready', scalable and high-performing environment currently available for sentiment analysis development. If you're willing to integrate external sentiment analysis modules into its core services, SpaCy could offer unrivaled speed benefits.

If you're willing to integrate external sentiment analysis modules into its core services, SpaCy could offer unrivaled speed benefits

3: TextBlob

Released:
2013
Created by:
Steven Loria
Highlights:
lightweight and accessible; rich sentiment analysis capabilities out of the box

Offering a greater ease-of-use and a less oppressive learning curve, TextBlob is an attractive and relatively lightweight Python 2/3 library for NLP and sentiment analysis development.

The project provides a more accessible interface compared to the capabilities of NLTK, and also leverages the Pattern web mining module from the University of Antwerp. Combining these resources makes it easy to switch between the capable Pattern library and, for example, a pre-trained NLTK model.

TextBlob has a rule-based integrated sentiment analysis function with two properties—subjectivity and polarity. Workflows with TextBlob and VADER (Valence Aware Dictionary and sEntiment Reasoner) are among the most popular approaches to sentiment analysis with TextBlob.

Given its design and goals, it's not surprising that TextBlob in itself has few functional characteristics to distinguish it from its competitors. It's capable and full-featured, but in terms of speed remains dependent on its external resources, neither of which are exemplary in this respect.

However, certain operations, such as extracting noun phrases, become notably less tortuous in TextBlob as compared to its rivals. It also provides a convenient native wrapper around the Google Translate API.

Things to Watch Out For

TextBlob expects ASCII text input by default, and could throw arcane errors if it doesn't get it. Therefore, your project may need a stratum of decode libraries or functions to keep the wheels moving.

If your workflow involves the processing of CSV files, it's worth observing that Unicode input isn't supported with TextBlob running on Python 2. If you're unable to switch to Python 3, your pipeline may need to convert CSVs into the UTF-8 format.

Since they're rolled into the package, the capabilities and limitations of Pattern are also a factor when evaluating TextBlob for our project. Pattern runs slower than SpaCy, for instance. You'll also need to check that TextBlob's native sentiment analysis functionality fits your project needs, and whether third-party libraries or modules are available to address any shortfall.

Final Considerations

So long as you consider the scope as well as the latency and scalability requirements of your project, TextBlob could be the quickest way to resolve a modular challenge in a larger pipeline.

Certain operations, such as extracting noun phrases, become notably less tortuous in TextBlob as compared to its rivals

4: Stanford CoreNLP

Released:
2010
Created by:
Stanford Natural Language Processing Group
Highlights:
platform-agnostic; multi-language support; a live demo available

Stanford CoreNLP is a highly extensible set of Java libraries for natural language analysis, which accesses Python via wrappers. It is platform-agnostic, feature-rich, efficient, and currently very popular in production systems.

CoreNLP offers good support for non-English languages in NLP flows. Current language models include Arabic, Chinese, French, German, and Spanish.

The suite is regularly updated and provides a wide variety of APIs for different programming languages. It has an efficient and stable annotator for arbitrary texts, as well as integration with annotation pipelines. Some of the CoreNLP components also support modules from NLTK.

CoreNLP comes with a native sentiment analysis tool, which has its own dedicated third-party resources. Stanford maintains a live demo with the source code of a sample sentiment analysis implementation.

Support is available through the stanford-nlp tag on Stack Overflow, as well as via mailing lists and support emails. Stanford's NLP mailing list archives are an additional resource.

Things to Watch Out For

Whether or not CoreNLP is fast seems to be in constant debate, and dependent on the scale, structure and setup of the implementation in question. The development team behind the system have acknowledged longstanding complaints about CoreNLP's speed as well as its occasional memory-usage issues.

Final Considerations

Its features, relative ease of implementation, dedicated sentiment analysis tools and good community support make CoreNLP a serious contender for production, even if its Java-based architecture could entail a little extra engineering and overhead, in some circumstances.

Its features, relative ease of implementation, dedicated sentiment analysis tools and good community support make CoreNLP a serious contender for production

5: Gensim

Released:
2010

Created by:
Radim Řehůřek, multiple contributors
Highlights:
scalable and speedy; strong native capabilities; commercial spinoffs available

Gensim originated from the work of two students at the Natural Language Processing Laboratory in the Czech Republic around 2010, and has matured into one of the most scalable and powerful options for NLP projects.

Like NLTK, Gensim is comprehensive and powerful enough to be used as a remote resource in wider pipelines—for instance, to provide assistance with phrase modeling, or to be utilized in tandem with other frameworks, such as SpaCy and TextaCy.

Gensim is a popular tool for topic and vector space modeling, and document similarity. It is also a strong resource for multi-label classification and dimensionality reduction. However, Gensim's primary focus is on the effective initial distillation of data from documents and word clouds.

Its native and highly optimized implementation of Google's word2vec machine learning models makes it a strong contender for inclusion in a sentiment analysis project, either as a core framework or as a library resource.

Gensim provides support for Cython implementations, offering SpaCy-like processing times, depending on the tasks at hand. In March 2019, the project released a new set of optimizations offering considerable speed boosts across a range of features.

Although the library is free to use, it's worth knowing that Gensim's originators have gone on to develop two similar commercial projects: the data analysis project PII Tools and the automated content analysis framework ScaleText—which the founders publicize as 'Gensim on steroids'. The original project, however, is well-maintained.

Besides the usual online tech communities, such as Stack Overflow, support for Gensim comes in the form of a dedicated Google Group or through professional consultation from one of the founders.

Final considerations

Gensim's tagline 'Topic Modeling for Humans' reveals both its advantages and limitations. As a highly-specialized and well-optimized set of Python NLP libraries, it's perhaps more likely to enter your sentiment analysis project as a facet rather than a base framework.

Like NLTK, Gensim is comprehensive and powerful enough to be used as a remote resource in wider pipelines

Afterword

In this round-up of some of the most popular NLP frameworks for Python sentiment analysis, we haven't had time to cover other strong contenders such as Polyglot, Scikit-learn, or MontyLingua. While we encourage your independent research, we are open to providing any further guidance in one-on-one consultation. Get in touch by filling in this simple form below.