SpaCy: What Is It and What Can It Do?

Unrelated to this course, I stumbled upon the spaCy Python package while working on a project for my job. Bankers working on loan invoices spend large portions their time parsing through documents to determine their purpose. With this information, they need to decide a certain course of action to fulfill documentation needs and client requests. This whole process proved inefficient, especially since technology could automate the process. As soon as I joined the company and arrived for my first day of work, I was tasked with solving this problem.

Diving further into the requirements for successfully completing the assignment, I learned that not only did I need to have the documents classified, I needed to parse out specific fields to return to the business. At the surface level, it appeared as though a few simple regex patterns could easily solve this problem. Everything changed when the test data samples came in.

Each client and partner sending us invoices used different templates. For a given client/vendor, the template for a document of a given class varied. The lack of standardizations resulted in a bunch of random invoices in random formats. To add more to the problem, many irrelevant fields also satisfy the regex patterns my team initially had in mind. As it turns out, the businesspeople working with these invoices every day became proficient at doing their job through years of accumulated experience. Even then, mistakes occur. If a human could learn these processes over time, could an algorithm be applied to capture this understanding and automate the process while committing fewer errors? Nobody in my team had knowledge on the matter, so I turned to the internet and encountered spaCy.

Surprisingly, I have never heard of spaCy before. Companies such as Airbnb, Uber, Quora, Microsoft, The Washington Post, and BBC have adopted its usage. What even is spaCy? Simply put, spaCy is a free and open-source library designed for advanced natural language processing in production environments. This means it may efficiently process large quantities of real-world text data. People can use it for creating systems for information extraction and natural language understanding. Furthermore, it has the capability of pre-processing text for deep learning algorithms and even train models.

With the high-level picture of spaCy out of the way, we now must understand what it tries to not do before diving deeper. spaCy cannot be considered as a platform or API and it is simply an installed library. The focus revolves mainly around getting things done for production. It is not meant for research as this allows it to limit the number of algorithms used, thereby allowing for more performance optimizations and ease of use. This, however, means the user is constrained to a few models, meaning users will unlikely be able to specify certain machine learning algorithms to create the most state-of-the-art cutting-edge natural language processing pipelines. Finally, the scope of spaCy limits itself within the realm of text processing and related machine learning algorithms. People who use spaCy must adapt its output for their own use-cases.

With that out of the way, we will now explore the features spaCy has to offer. Without going too much in detail, the text processing capabilities spaCy mainly focuses on are tokenization, part-of-speech tagging, dependency parsing, lemmatization, sentence boundary detection, named entity recognition and entity linking. Using these natural language processing concepts, spaCy may then do text classification, similarity analysis, rule-based matching, and customized model training. Each spaCy object may be saved via a built-in serialization feature.

Straight out of the box, spaCy provides a vast array of functionalities. It comes with built in dictionaries for mapping words during lemmatization as well as text processers trained on various corpuses. The user simply selects which training corpus the NLP model originated from

to load in the respective processor. Naturally, these models tend to have bias leaning towards a generalized corpus, causing problems for more niche applications involving documents containing specialized vocabulary and linguistic patterns. Lucky for us, the authors foresaw such issues and provided us with methods to create our own customizable processors.

To create such models, we would be required to have training data. Unfortunately, this means that somebody must go out and manually parse out collections of documents to feed into spaCy. To make matters worse, if your text processing pipeline requires multiple components such as a part-of-speech tagger and a named entity recognizer, you'll need training, evaluation and test data for each. spaCy does have built in standards and functions to help facilitate the process, but at the end of the day, somebody must do some manual labor. Starting from version 2.1 onwards, spaCy added a transfer learning feature to help increase training efficiency and model accuracy. Although creating training data could prove to be a grueling endeavor, a positive side-effect of these training procedures allows for extra customizations to be added. For example, the default named entity recognizer has a preset number of distinct categories. By adding in your own training data, you may also specify a new entity such as an ID entity. Additionally, after the training process these models may be saved to disk and easily loaded in afterwards for usage or further training via transfer learning.

Once satisfied with the text processing components, the user can easily create a natural language processing pipeline to consume text and spit out the corresponding processed document for whatever the use-case may be. In case a user wonders how the processed data looks like, spaCy has built in visualizers for syntax and named entity recognition.

If, however, the user can be satisfied by general models, they have several options to choose from. In fact, 46 statistical models have been created for 16 languages. spaCy currently supports over 61 languages. Being only five years old with an active Github repository, more performance enhancing updates and language support will likely come over time.

Of course, with this being a brief overview of spaCy, many more nuances and features are available to be explored. The ones mentioned only highlight the most basic functionalities in order to capture the usefulness of the package. Compared to similar packages, spaCy offers little variety in algorithms for user selection, but delivers high quality results efficiently. I would recommend this package for people who simply need to get text processing and classifications tasks done without worrying about the nitty-gritty details. The package is easy to use and well-documented, allowing users quickly to hit the ground running on new projects.

After discovering spaCy, I realized I could turn my problem of finding specific fields into a classification problem. Given a document, determine whether a string of characters is the field in question. In other words, I simply needed to train a named entity recognizer to look for the fields constrained to the fact that each document can only contain one such field of each category. However, instead of being interested in just the class of the labeled fields, we want to also extract the field itself. Finding the answer to one problem gives us the answer to another. Creating the training dataset manually on a few hundred documents took around two weeks. Although a few hundred seems like a small number, the resulting named entity recognizer outperformed regex statements for every field, getting results close to 80% accuracy. To clarify, the document classification task and field extract tasks were done via different means. The task highlighted in this paragraph is just the field extraction task remade into a classification problem.

With spaCy's ease of use, efficiency, and ability to produce high quality results, it's no wonder many companies support it. the next time a text processing task arises for an industrial purpose, keep spaCy in mind.

Resources:

spaCy main website: https://spacy.io/
spaCy Github page: https://github.com/explosion/spaCy