Overview of Named Entity Recognition

UIUC | CS 410: Text Information Systems | Fall 2022
Ethan Alberto (ethanma3)

The purpose of this review is to briefly describe the concept of Named Entity Recognition (NER) as a key component of natural language processing (NLP), understand how it works and introduce some options of open source packages available, that would enable you to to quickly perform NER on a given document set.

1 Introduction

In order to discuss Named Entity Recognition (NER), it is worth defining an 'entity'. Entities are considered to be the most important chunks of a sentence such as noun phrases, verb phrases or both. NER can now be defined as a task that simply scans entire texts and extracts these fundamental entities and classifies them into predefined semantic categories such as *person (PER), location (LOC), geo-political entity (GPE).*

2 Understanding Named Entity Recognition (NER)

2.1 Working of NER

The use-case plays an important role in defining the entities you might be interested in. NER works based on it, but the main goal is to extract crucial information of all entities in a given document. Entity recognition, processes structured and unstructured documents by identifying and locating these entities. The best way of depicting the working of NER works is through an example; Instead of recognising "Steve" and "Jobs" as separate entities, NER understands that "Steve Jobs" is actually a single entity. More advanced NER algorithms go one step further to even classify entities into predefined class. This means that NER is not only able to identify "Steve Jobs" as an entity but also classify it as a person.

2.2 Process of NER

For the process of NER, you need to build a Knowledge Base (KB) which will contain the known named entities. You then try to link an entity to the knowledge base entity node if it exists. This process can be separated into three main components: extractors, searchers and disambiguators:

1. **Extractor:**
   Extraction is the detection and preparation of named entity mentions. The extraction phase includes part of speech (POS) tagging, tokenisation, sentence boundary detection, capitalisation rules and in-document co-referencing. In-document co-reference, in particular, is important and is used to find more specific search terms.

2. **Searcher:**
   Search is the process of generating a set of candidate KB entities for a mention. Titles, disambiguation pages and other Wikipedia-derived aliases can be leveraged at this stage to capture synonyms. A searcher should balance precision and recall to capture the correct entity while maintaining a small set of candidates in order to reduce the computation required for disambiguation.

   The goal of the searcher is to extract a set of aliases — potential mention strings that can refer to an entity, for each article. By querying an index over these aliases, candidate referents for each entity mention are found.

A way for achieving improved disambiguation efficiency is by reducing the number of candidates that must be considered. Bearing in mind that the correct candidate is not always the first one returned by the searcher.

During the search phase the system proposes a set of candidates for a named entity mention to be linked to, which are then ranked by the disambiguator.

3. **Disambiguator:**
In disambiguation, the best entity is selected for a mention.

3 Tools for Named Entity Recognition

According to [AL13], NER is key in recognizing and extracting data, which is a fundamental task and a core process of the natural language processing field (NLP), mainly for two reasons. First, NER is used directly in many applied research domains [Nad07]. For instance, proteins and genes can be considered as named entities, and many works in medicine focus on the analysis of scientific articles to find out hidden relationships between them, and drive experimental research [TXT+05]. But NER is also used as a pre-processing step by more advanced NLP tools, such as relationship or information extraction [BB07]. As a result, a number of tools have been developed to perform NER.
These NER tools differ in many ways. First, they vary in the methods used, this could range from completely manually specified systems (e.g. grammar rules) to fully automatic machine-learning processes, this could even include hybrid approaches that combine the best of both worlds. Second, they do not necessarily handle the same classes of entities. Third, some of these tools are generic and can be applied to any text [RR09, Lan11, FGM05], while others might be domain-specific such as bio-medicine [LG08] or geography [MCS05]. Fourth, some are implemented as libraries [FGM05], and some take various other forms such as Web services [1]. Fifth, the data outputted by NER tools can take various forms, usually programmatic objects for libraries and text files for the others. There is no standard for files containing NER processed text, so output files can vary a lot from one tool to the other. Sixth, tools reach different levels of performance. Moreover, their accuracy can vary depending on the considered type of entity, class of text, etc.

For this review, we will focus on a few open-source NER tools available:

1. **SpaCy[2]:**
SpaCy is an open-source NLP library that can be used for various tasks and to process large amounts of data. With support for over 64 languages and 63 trained pipelines for 19 languages, it is a handy tool in NLP. Under the hood, SpaCy uses Bloom embeddings and residual CNN's to identify named entities.

2. **Stanford NER[3]:**
Stanford NER [FGM05] is a Java implementation of a named entity recogniser. Named Entity Recognition (NER) labels sequences of words in a text which are the names of things, such as person and company names, or gene and protein names. The standard download of the model includes good recognisers for the English language and works particularly well for 3 classes (NAMES, ORGAISATION and LOCATION). It also comes with models for 4 and 7 classes as well as provides support in languages like German, Spanish and Chinese. Stanford NER has up-to-date interfaces for computer languages like Apache Tika, JavaScript, C#, Python, Ruby, etc to name a few.

3. **NLTK[4]:**
Natural language Toolkit (NLTK) is a set of libraries used for NLP. It is widely used in research and for educational purposes. Written in Python, it has access to more than 50 text corpora across 7 languages. One of the primary principles of NLTK, besides simplicity, consistency, and extensibility, is modularity. Modularity provides components that can be used independently.

[1] Thomson Reuters. (2008). Calais Web Service. Available: http://www.opencalais.com/
[2] https://spacy.io/
[3] https://nlp.stanford.edu/software/CRF-NER.shtml
[4] https://www.nltk.org/

This might be especially useful for tuning only specific parts of the pipeline or even using third parties in conjunction with this toolkit. NLTK does NER in two steps. The first step is POS (parts-of-speech) tagging or grammatical tagging, which is followed by chunking to extract the named entities.

4. **Flair**[5]**:**
   Flair [ABV18, ABB+19] is a simple framework developed and made open-source by Zalando Research, for state-of-the-art NLP . Flair is built on top of PyTorch, which is a powerful deep learning framework. Claimed to support over 275 languages, it is very useful in training small models. It is pre-trained on an extremely large unlabelled text corpora. Flair has a few features that tend to give it an edge over other NLP libraries. First, it comprises of popular and state-of-the-art word embeddings, such as GloVe, BERT, ELMo, Character Embeddings, etc. These are very easy to use thanks to the Flair API. Second, Flair's interface allows us to combine different word embeddings and use them to embed documents. This in turn leads to a significant uptick in results. Third, 'Flair Embedding' is the signature embedding provided within the Flair library. It is powered by contextual string embeddings and finally, it has support for a number of languages and is always looking make new additions.

4 Conclusion

Based on the libraries and toolkits described above, there is no metric that can clearly point out a clear winner or show that one of them works better than the other. However, each of them have has their unique advantages and picking a package is purely use-case driven. While it is clear that Flair and SpaCy perform better than NLTK, a choice between SpaCy and Flair can be taken, considering the nature of the dataset. For smaller datasets, Flair is a good option considering the accuracy. The time taken will be almost the same as that of SpaCy. SpaCy compromises its precision for its speed. Stanford NER is slow for NLP production usage, but can integrate with NLTK to boost CPU efficiency. NLTK is slower when compared to the other packages. It doesn't employ neural networks and splits the sentences without considering the semantics of the sentences. SpaCy is huge, however, the package is not customisable and the internals are not opaque. Flair is more ideal for smaller applications but its lower speed makes SpaCy a better candidate for large datasets.

References

[ABB+19] Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. FLAIR: An easy-to-use framework for state-of-the-art NLP. In *NAACL 2019, 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 54–59, 2019.

[ABV18] Alan Akbik, Duncan Blythe, and Roland Vollgraf. Contextual string embeddings for sequence labeling. In *COLING 2018, 27th International Conference on Computational Linguistics*, pages 1638–1649, 2018.

[AL13] Samet Atdağ and Vincent Labatut. A comparison of named entity recognition tools applied to biographical texts. In *2nd International conference on systems and computer science*, pages 228–233. IEEE, 2013.

[BB07] Nguyen Bach and Sameer Badaskar. A review of relation extraction. *Literature review for Language and Statistics II*, 2:1–15, 2007.

[FGM05] Jenny Rose Finkel, Trond Grenager, and Christopher D Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd annual meeting of the association for computational linguistics (ACL'05)*, pages 363–370, 2005.

[5] https://github.com/flairNLP/flair

[Lan11] F Landsbergen. Named entity work in impact. In *IMPACT final conference*, volume 2011, 2011.

[LG08] Robert Leaman and Graciela Gonzalez. Banner: an executable survey of advances in biomedical named entity recognition. In *Biocomputing 2008*, pages 652–663. World Scientific, 2008.

[MCS05] Bruno Martins, Marcirio Chaves, and Mário J Silva. Assigning geographical scopes to web pages. In *European Conference on Information Retrieval*, pages 564–567. Springer, 2005.

[Nad07] David Nadeau. *Semi-supervised named entity recognition: learning to recognize 100 entity types with little supervision*. PhD thesis, University of Ottawa (Canada), 2007.

[RR09] Lev Ratinov and Dan Roth. Design challenges and misconceptions in named entity recognition. In *Proceedings of the thirteenth conference on computational natural language learning (CoNLL-2009)*, pages 147–155, 2009.

[TXT+05] Lorraine Tanabe, Natalie Xie, Lynne H Thom, Wayne Matten, and W John Wilbur. Genetag: a tagged corpus for gene/protein named entity recognition. *BMC bioinformatics*, 6(1):1–7, 2005.