Assignment 2, addendum

Matko Bošnjak

November 12, 2014

The purpose of this document is to give more details about the dataset for the second assignment, provide explanations to some of the preprocessing executed on it, and point to additional analysis tools.

1 Dataset origin

The dataset that you've been handed is a preprocessed version of the BioNLP 2011 Genia Task. Should you require, you can find more details about the (original) dataset at the following webpage:

`http://2011.bionlp-st.org/home/genia-event-extraction-genia`

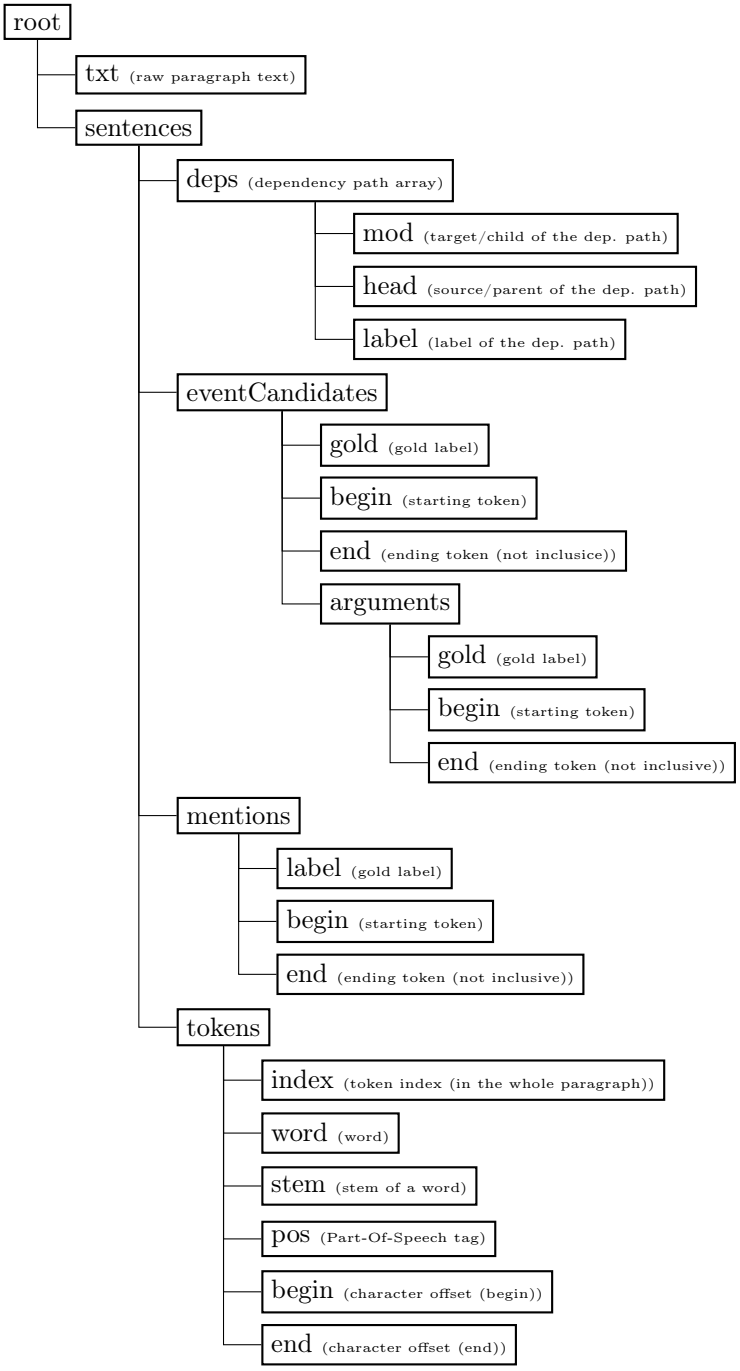
The dataset available on the task webpage is exported in a different format, and does not contain any preprocessing (sentence splitting, tokenization, dependency path extraction).

2 Dataset format

2.1 JSON files

The data you downloaded consists of a number of `.json` files, written in JSON (JavaScript Object Notation)[1] format. Each file contains a document/paragraph that specifies the text, an array of sentences, which contain tokens, dependency path, mentions, and event candidates. The complete structure of these JSON files is the following:

[1] If you never encountered this format before, you can find more details at `http://www.json.org/`



You might find it very useful to view the JSON files with a dedicated JSON viewer, like one of the following:

`http://jsonviewer.stack.hu/`
`http://www.jsoneditoronline.org/`

Or, if you're using Chrome as your browser, you can install a JSONView plugin[2]

`https://chrome.google.com/webstore/detail/jsonview/chklaanhfefbnpoihckbnefhakgolnmc`[3]

2.2 WW files

We will not be using ww files for data loading. However, they can be very useful for two reasons. First, you can visualize them with the "What's wrong with My NLP?", found there:

`https://code.google.com/p/whatswrong/`

You can find details on using `whatswrong` in the Assignment 2 document.

The second use of ww files is that they are a bit easier to inspect from a regular text editor. In total, these files convey the same information as the JSON files.

3 Dataset preprocessing

The dataset comes preprocessed. Specifically, we already tokenized the text and split sentences. Novelty over here is the addition of Part-Of-Speech (POS) tags and dependency paths.

Examples covered in the rest of the text are coming from the first sentence in the `PMID-1313226.json` file. The visualizations presented are produced with the Stanford CoreNLP [1] webpage, with the integrated Brat tool [3]. Please check those tools as they could be of great assistance when choosing features and visualizing your models' results.

The Brat tool is available here:

`http://brat.nlplab.org/`

and the Stanford CoreNLP webpage, showcasing a couple of tools in the CoreNLP package (including POS tagging and dependency path extraction), can be found here:

`http://nlp.stanford.edu:8080/corenlp/`

[2] In case you'll be using this plugin, you need to go to extension settings and allow the plugin to access file URLs

[3] You need to allow the plugin to access file URLs under Settings > Extensions > JSONView

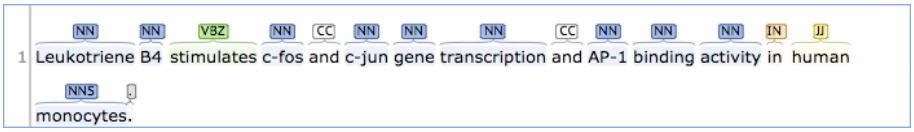
NOTE: We advise you to use "What's Wrong With My NLP?" visualizer as you already have `.ww` files necessary for visualization with it. Preprocessing and thus visualizations done by the Stanford CoreNLP webpage do not correspond fully to the training set! The next version of this document will be switched to use the "What's Wrong With My NLP?" visualizer.

3.1 Part-Of-Speech tagging

You will find the POS tag as the `pos` property for each token. These tags have been extracted by a POS Tagger[4], a tool that assigns parts of speech (word categories) to each word/token. Examples of such tags are noun, verb, adjective, etc. The POS tags used in this assignment are more fine-grained POS tags, corresponding to Penn Treebank English [2] POS tag set, containing 36 categories in total. You can find the complete list of these tags here:

`https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html`

The output of the Stanford CoreNLP POS tagger on our running example is:



You can easily read off POS tags from their respective tokens, e.g. `Leukotriene` is labeled as `NN`, meaning noun, singular or mass, `stimulates` is labeled as `VBZ`, a verb in 3rd person singular present, `human` is labeled as `JJ`, an adjective, etc.

Note that there might be some discrepancies between the results on the Stanford CoreNLP webpage and the data in the dataset due to the fact that we used one of the previous versions of the CoreNLP package.

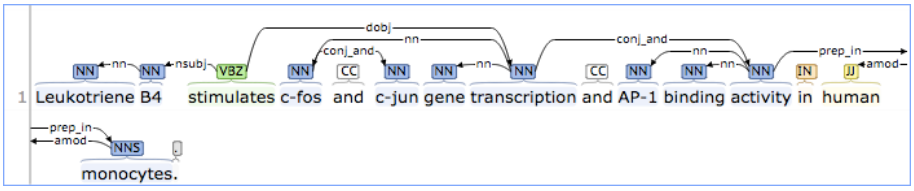
3.2 Dependency path

The data files also contain a list of syntactic dependencies that can be useful as features, in a form of dependency path/parse/tree. Dependency path/parse/tree is a typed representation of grammatical dependencies designed to provide a simple illustration of grammatical relationship in a sentence. It represents sentence relationships uniformly as typed dependency relations. Specifically, they are represented as triples: a relation between a pair of words, encoded as head word (`head`), modified word (`mod`) and the label of the relation (`label`). This representation is very useful across a range of NLP usages. There are around 50 grammatical relation in total. for more detail on the dependency path, take a look at the following document:

[4]Stanford CoreNLP POS Tagger, to be precise

`http://nlp.stanford.edu/software/dependencies_manual.pdf`

The output of the Stanford CoreNLP dependency parser (collapsed version), on our running example yields:



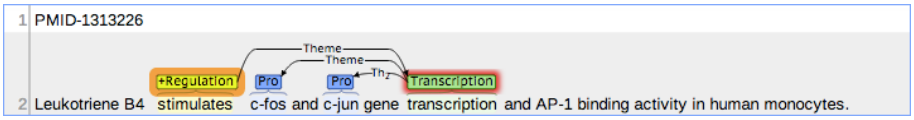
In this example, we can see that `B4` is a nominal subject (`nsubj`) of `stimulates` (a noun phrase which is the syntactic subject of a clause). `transcription` and `activity` are in a conjunctive and relation (`conj_and`), `human` is an adjectival modifier (`amod`) of `monocytes`, and `monocytes` and `activity` are in a prepositional modifier (`prep_in`) relation, over the preposition `in`, etc.

Note that there might be (precisely, there will be) some discrepancies between the dependency path provided in the dataset and the one shown on the webpage, due application of a prior version of the Stanford CoreNLP dependency parse on the dataset.

4 Dataset labels

You will find protein mentions with their labels under the field `mentions`, and a list of candidate event triggers, under `eventCandidates`. Each event candidate will have a list of candidate argument edges under the field `arguments`. True triggers and arguments are labeled with their respective gold labels (`gold`), and the rest of them are labeled with `None` (negative examples).

The output of Brat, visualizing `eventCandidates` and `mentions` field of our running example would look like this:



Note that there will be many event candidates and their arguments labeled as None.

5 Dataset loading and data structure

You can load the data using the provided `LoadData` object in the `uk.ac.ucl.cs.mr.assignment2` package. This object loads the JSON format into the following structure (un-

derlined words denote variables):

```
Document(txt, wolfeSentences)
wolfeSentences = Sentence(wolfeTokens,
                    SyntaxAnnotation(null, wolfeDependencyTree),
                    IEAnnotation(wolfeMentions, Nil, wolfeEventCandidates))
wolfeTokens = Token(word, CharOffsets(begin, end), pos, stem)
wolfeDependencyTree = DependencyTree(wolfeTokens, wolfeDependencyTreeArcs)
wolfeDependencyTreeArcs = (mod, head, label)
wolfeMentions = EntityMention(label, begin, end)
wolfeEventCandidates = EventMention(``eventCandidate'',
                                EntityMention(``gold'', begin, end),
                                arguments)
arguments = RoleMention(``argument'',
                  EntityMention(``gold'', begin, end))
```

References

[1] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, 2014.

[2] Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330, 1993.

[3] Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. Brat: a web-based tool for nlp-assisted text annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 102–107. Association for Computational Linguistics, 2012.