MSR SPLAT, a language analysis toolkit

Chris Quirk, Pallavi Choudhury, Jianfeng Gao, Hisami Suzuki, Kristina Toutanova, Michael Gamon, Wen-tau Yih, Lucy Vanderwende
Microsoft Research
Redmond, WA 98052 USA
{chrisq, pallavic, jfgao, hisamis, kristout, mgamon,scottyih, lucyv@microsoft.com}

Colin Cherry
National Research Council Canada
1200 Montreal Road
Ottawa, Ontario K1A 0R6
colin.cherry@nrccnrc.gc.ca

Abstract

We describe MSR SPLAT, a toolkit for language analysis that allows easy access to the linguistic analysis tools produced by the NLP group at Microsoft Research. The tools include both traditional linguistic analysis tools such as part-of-speech taggers, constituency and dependency parsers, and more recent developments such as sentiment detection and linguistically valid morphology. As we expand the tools we develop for our own research, the set of tools available in MSR SPLAT will be extended. The toolkit is accessible as a web service, which can be used from a broad set of programming languages.

1 Introduction

The availability of annotated data sets that have become community standards, such as the Penn TreeBank (Marcus et al., 1993) and PropBank (Palmer et al., 2005), has enabled many research institutions to build core natural language processing components, including part-of-speech taggers, chunkers, and parsers. There remain many differences in how these components are built, resulting in slight but noticeable variation in the component output. In experimental settings, it has proved sometimes difficult to distinguish between improvements contributed by a specific component feature from improvements due to using a differently-trained linguistic component, such as tokenization. The community recognizes this difficulty, and shared task organizers are now providing accompanying parses and other analyses of the shared task data. For instance, the BioNLP shared task organizers have provided output from a number of parsers[1], alleviating the need for participating systems to download and run unfamiliar tools. On the other hand, many community members provide downloads of NLP tools[2] to increase accessibility and replicability of core components.

Our toolkit is offered in this same spirit. We have created well-tested, efficient linguistic tools in the course of our research, using commonly available resources such as the PTB and PropBank. We also have created some tools that are less commonly available in the community, for example linguistically valid base forms and semantic role analyzers. These components are on par with other state of the art systems.

We hope that sharing these tools will enable some researchers to carry out their projects without having to re-create or download commonly used NLP components, or potentially allow researchers to compare our results with those of their own tools. The further advantage of designing MSR SPLAT as a web service is that we can share new components on an on-going basis.

2 Parsing Functionality

2.1 Constituency Parsing

[1] See www-tsujii.is.s.u-tokyo.ac.jp/GENIA/SharedTask for the description of other resources made available in addition to the shared task data.
[2] See, for example, http://nlp.stanford.edu/software; http://www.informatics.sussex.ac.uk/research/groups/nlp/rasp; http://incubator.apache.org/opennlp

The syntactic parser in MSR SPLAT attempts to reconstruct a parse tree according the Penn TreeBank specification (Marcus et al., 1993). This representation captures the notion of labeled syntactic constituents using a parenthesized representation. For instance, the sentence "Colorless green ideas sleep furiously." could be assigned the following parse tree, written in the form of an S expression:

```
(TOP (S
   (NP (JJ Colorless) (JJ green) (NNS ideas))
   (VP (VB sleep) (ADVP (RB furiously)))
   (. .)))
```

For instance, this parse tree indicates that "Colorless green ideas" is a noun phrase (NP), and "sleep furiously" is a verb phrase (VP).

Using the Wall Street Journal portion of the Penn TreeBank, we estimate a coarse grammar over the given grammar symbols. Next, we perform a series of refinements to automatically learn fine-grained categories that better capture the implicit correlations in the tree using the split-merge method of Petrov et al. (2006). Each input symbol is split into two new symbols, both with a new unique symbol label, and the grammar is updated to include a copy of each original rule for each such refinement, with a small amount of random noise added to the probability of each production to break ties. We estimate new grammar parameters using an accelerated form of the EM algorithm (Salakhutdinov and Roweis, 2003). Then the lowest 50% of the split symbols (according to their estimated contribution to the likelihood of the data) are merged back into their original form and the parameters are again re-estimated using AEM. We found six split-merge iterations produced optimal accuracy on the standard development set.

The best tree for a given input is selected according to the max-rule approach (cf. Petrov et al. 2006). Coarse-to-fine parsing with pruning at each level helps increase speed; pruning thresholds are picked for each level to have minimal impact on development set accuracy. However, the initial coarse pass still has runtime cubic in the length of the sentence. Thus, we limit the search space of the coarse parse by closing selected chart cells before the parse begins (Roark and Hollingshead, 2008). We train a classifier to determine if constituents may start or end at each position in the sentence. For instance, constituents seldom end at the word "the" or begin at a comma. Closing a number of chart cells can substantially improve runtime with minimal impact on accuracy.

2.2 Dependency Parsing

The dependency parses produced by MSR SPLAT are unlabeled, directed arcs indicating the syntactic governor of each word.

These dependency trees are computed from the output of the constituency parser. First, the head of each non-terminal is computed according to a set of rules (Collins, 1999). Then, the tree is flattened into maximal projections of heads. Finally, we introduce an arc from a parent word *p* to a child word *c* if the non-terminal headed by *p* is a parent of the non-terminal headed by *c*.

2.3 Semantic Role Labeling

The Semantic Role Labeling component of MSR SPLAT labels the semantic roles of verbs according to the PropBank specification (Palmer et al., 2005). The semantic roles represent a level of broad-coverage shallow semantic analysis which goes beyond syntax, but does not handle phenomena like co-reference and quantification.

For example, in the two sentences "John broke the window" and "The window broke", the phrase *the window* will be marked with a THEME label. Note that the syntactic role of the phrase in the two sentences is different but the semantic role is the same. The actual labeling scheme makes use of numbered argument labels, like ARG0, ARG1, …, ARG5 for core arguments, and labels like ARGM-TMP,ARGM-LOC, etc. for adjunct-like arguments. The meaning of the numbered arguments is verb-specific, with ARG0 typically representing an agent-like role, and ARG1 a patient-like role.

This implementation of an SRL system follows the approach described in (Xue and Palmer, 04), and includes two log-linear models for argument identification and classification. A single syntax tree generated by the MSR SPLAT split-merge parser is used as input. Non-overlapping arguments are derived using the dynamic programming algorithm by Toutanova et al. (2008).

3 Other Language Analysis Functionality

3.1 Sentence Boundary / Tokenization

This analyzer identifies sentence boundaries and breaks the input into tokens. Both are represented as offsets of character ranges. Each token has both a raw form from the string and a normalized form in the PTB specification, e.g., open and close parentheses are replaced by -LRB- and -RRB-, respectively, to remove ambiguity with parentheses indicating syntactic structure. A finite state machine using simple rules and abbreviations detects sentence boundaries with high accuracy, and a set of regular expressions tokenize the input.

3.2 Stemming / Lemmatization

We provide three types of stemming: Porter stemming, inflectional morphology and derivational morphology.

3.2.1 Stems

The stemmer analyzer indicates a stem form for each input token, using the standard Porter stemming algorithm (Porter, 1980). These forms are known to be useful in applications such as clustering, as the algorithm assigns the same form "dai" to "daily" and "day", but as these forms are not citation forms of these words, presentation to end users is known to be problematic.

3.2.2 Lemmas

The lemma analyzer uses inflectional morphology to indicate the dictionary lookup form of the word. For example, the lemma of "daily" will be "daily", while the lemma of "children" will be "child". We have mined the lemma form of input tokens using a broad-coverage grammar NLPwin (Heidorn, 2000) over very large corpora.

3.2.3 Bases

The base analyzer uses derivational morphology to indicate the dictionary lookup form of the word; as there can be more than one derivation for a given word, the base type returns a list of forms. For example, the base form of "daily" will be "day", while the base form of "additional" will be "addition" and "add". We have generated a static list of base forms of tokens using a broad-coverage grammar NLPwin (Heidorn, 2000) over very large corpora. If the token form has not been observed in those corpora, we will not return a base form.

3.3 POS tagging

We train a maximum entropy Markov Model on part-of-speech tags from the Penn TreeBank. This optimized implementation has very high accuracy (over 96% on the test set) and yet can tag tens of thousands of words per second.

3.4 Chunking

The chunker (Gao et al., 2001) is based on a Cascaded Markov Model, and is trained on the Penn TreeBank. With state-of-the-art chunking accuracy as evaluated on the benchmark dataset, the chunker is also robust and efficient, and has been used to process very large corpora of web documents.

4 The Flexibility of a Web Service

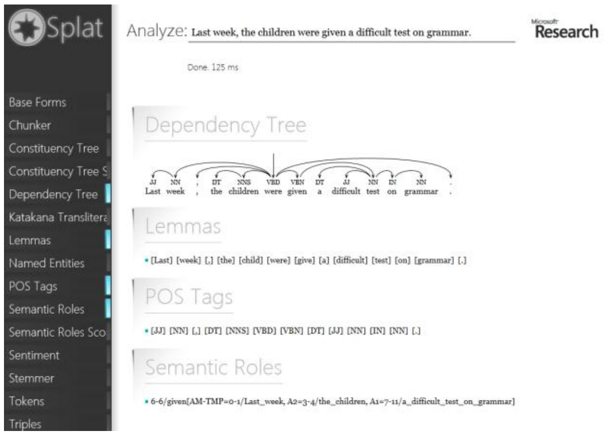
By making the MSR SPLAT toolkit available as a web service, we can provide access to new tools, e.g. sentiment analysis. We are in the process of building out the tools to provide language analysis for languages other than English. One step in this direction is a tool for transliterating between English and Katakana words. Following Cherry and Suzuki (2009), the toolkit currently outputs the 10-best transliteration candidates with probabilities for both directions.

Another included service is the Triples analyzer, which returns the head of the subject, the verb, and the head of the object, whenever such a triple is encountered. We found this functionality to be useful as we were exploring features for our system submitted to the BioNLP shared task.

5 Programmatic Access

5.1 Web service reference

We have designed a web service that accepts a batch of text and applies a series of analysis tools to that text, returning a bag of analyses. This main web service call, named "Analyze", requires four parameters: the language of the text (such as "en" for English), the raw text to be analyzed, the set of analyzers to apply, and an access key to monitor and, if necessary, constrain usage. It returns a list of analyses, one from each requested analyzer, in a



Analyze: Last week, the children were given a difficult test on grammar.

Done. 125 ms

• [Last] [week] [,] [the] [child] [were] [give] [a] [difficult] [test] [on] [grammar] [.]

• [JJ] [NN] [,] [DT] [NNS] [VBD] [VBN] [DT] [JJ] [NN] [IN] [NN] [.]

• 6-6/given[AM-TMP=0-1/Last_week, A2=3-4/the_children, A1=7-11/a_difficult_test_on_grammar]

Figure 1. Screenshot of the MSR SPLAT interactive UI showing selected functionalities which can be toggled on and off. This is the interface that we propose to demo at NAACL.

simple JSON (JavaScript Object Notation) format easy to parse in many programming languages.

In addition, there is a web service call "Languages" that enumerates the list of available languages, and "Analyzers" to discover the set of analyzers available in a given language.

5.2 Data Formats

We use a relatively standard set of data representations for each component. Parse trees are returned as S expressions, part-of-speech tags are returned as lists, dependency trees are returned as lists of parent indices, and so on. The website contains an authoritative description of each analysis format.

5.3 Speed

Speed of analysis is heavily dependent on the component involved. Analyzers for sentence separation, tokenization, and part-of-speech tagging process thousands of sentences per second; our fastest constituency parser handles tens of sentences per second. Where possible, the user is encouraged to send moderate sized requests (perhaps a paragraph at a time) to minimize the impact of network latency.

6 Conclusion

We hope that others will find the tools that we have made available as useful as we have. We encourage people to send us their feedback so that we can improve our tools and increase collaboration in the community.

7 Script Outline

The interactive UI (Figure 1) allows an arbitrary sentence to be entered and the desired levels of analysis to be selected as output. As there exist other such toolkits, the demonstration is primarily aimed at allowing participants to assess the quality, utility and speed of the MSR SPLAT tools. http://research.microsoft.com/en-us/projects/msrsplat/

References

Colin Cherry and Hisami Suzuki. 2009. Discriminative substring decoding for transliteration. In *Proceedings of EMNLP*.

Michael Collins. 1999. *Head-driven statistical models for natural language parsing*. PhD Dissertation, University of Pennsylvania.

Jianfeng Gao, Jian-Yun Nie, Jian Zhang, Endong Xun, Ming Zhou and Chang-Ning Huang. 2001. Improving query translation for CLIR using statistical Models. *In Proceedings of SIGIR*.

George Heidorn. 2000. Intelligent writing assistance. In R. Dale, H. Moisl and H. Somers (eds.), *A Handbook of Natural Language Processing: Techniques and Applications for the Processing of Text*. New York: Marcel Dekker.

Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics* 19(2): 313-330.

Martha Palmer, Dan Gildea, Paul Kingsbury. 2005. The Proposition Bank: An Annotated Corpus of Semantic Roles. *Computational Linguistics*, 31(1): 71-105

Martin Porter. 1980. An algorithm for suffix stripping. *Program*, 14(3): 130-137.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning Accurate, Compact, and Interpretable Tree Annotation. In *Proceedings of ACL*.

Brian Roark and Kristy Hollingshead. 2008. Classifying chart cells for quadratic complexity context-free inference. In *Proceedings of COLING*.

Ruslan Salakhutdinov and Sam Roweis. 2003. Adaptive Overrelaxed Bound Optimization Methods. In *Proceedings of ICML*.

Kristina Toutanova, Aria Haghighi, and Christopher D. Manning. 2008. A global joint model for semantic role labeling, *Computational Linguistics*, 34(2): 161-191.

Nianwen Xue and Martha Palmer. 2004. Calibrating Features for Semantic Role Labeling. In *Proceedings of EMNLP*.

Munmun de Choudhury, Scott Counts, Michael Gamon. Not All Moods are Created Equal! Exploring Human Emotional States in Social Media. Accepted for presentation in *ICWSM 2012*

Munmun de Choudhury, Scott Counts, Michael Gamon. Happy, Nervous, Surprised? Classification of Human Affective States in Social Media. Accepted for presentation (short paper) in *ICWSM 2012*