

Sentence and Token Splitting Based On Conditional Random Fields

Katrin Tomanek **Joachim Wermter** **Udo Hahn**
Jena University Language & Information Engineering (JULIE) Lab
Fürstengraben 30
D-07743 Jena, Germany
{tomanek|wermter|hahn}@coling-uni-jena.de

Abstract

Natural language processing systems which deal with real-world documents require several low-level tasks such as splitting a text into its constituent sentences, and splitting each sentence into its constituent tokens. These basic text segmentation services are usually supplied by some preprocessor prior to linguistic analysis. While this task is often considered as unsophisticated clerical work, in the natural sciences and engineering domains it poses particular problems due to complex naming conventions. In this paper, we first introduce an annotation framework for sentence and token splitting underlying a newly constructed sentence- and token-tagged biomedical text corpus. This corpus serves as a training environment and test bed for our machine-learning based sentence and token splitters using Conditional Random Fields (CRFs). Our evaluation experiments reveal that CRFs with a rich feature set substantially increase the performance for sentence and token segmentation in scientific documents, *viz.* from the biomedical domain.

1 Introduction

Natural language processing systems expect their input to be properly delimited into consecutive sentences and sentences to be properly segmented into their constituent tokens. Real-world documents lack such fine structure and, typically, simple heuristic pre-processors perform these segmentation tasks. In the newspaper domain (Grefenstette and Tapanainen, 1994), manually defined patterns for sentence splitting rely on the fact that, e.g., a period is a sentence boundary, if it

is followed by an uppercase letter and not part of a known abbreviation ('e.g.', 'Mr.', etc.). Tokenization is often performed by splitting at a closed set of special characters (especially punctuation symbols, quotation marks, parentheses, brackets, etc.). While such rules may be adequate for the newspaper domain, they are underspecified when applied to the biomedical domain. Here, complex naming conventions lead, in particular, to unsatisfactory tokenization results when few and simple hand-written rules are used. Both sentence and token boundary symbols are much more ambiguous because they often appear within entity names (organism, protein, cell names, etc.) and their abbreviations, as well as within (chemical) formulae, bibliographic references, etc. (Grover et al., 2006). If these challenges are not accounted for, errors in early processing steps will unavoidably be propagated upwards in an NLP analysis pipeline. As a consequence, text mining modules, such as named entity recognition and relation detection, which are fed by erroneous segmentation results will inevitably suffer in terms of performance.

Rather than adding further complexity to manually maintained rule sets, supervised machine learning (ML) is becoming more and more the method of choice for many demanding NLP tasks. From a given set of training examples a statistical model is automatically learned which is then used to assign labels to unseen data. Those approaches are to a large extent data-driven, i.e., by exchanging the training material they can be ported to other domains and languages, possibly without further changes. This constitutes a clear advantage over rule-based approaches which require a lot of manual tweaking and tuning, especially when more complex rules have to be supplied (Ngai and Yarowsky, 2000). Furthermore, supervised ML approaches have been shown to outperform rule-based ones on several NLP tasks,

including sentence boundary detection and tokenization, both in performance and breadth of coverage (Palmer and Hearst, 1997).

Therefore, we developed a new approach to sentence and token splitting based on Conditional Random Fields (CRFs), a sequential machine learning technique (Lafferty et al., 2001). Conditional Random Fields have recently been applied to similar segmentation tasks, such as e.g. word segmentation in Chinese texts (Peng et al., 2004) and sentence boundary detection in speech (Liu et al., 2005). However, to the best of our knowledge, it has not been shown before how CRFs can be applied to sentence splitting and tokenization in scientific documents from the biomedical domain.

For sentence and token boundary detection in the life sciences domain, two annotated text corpora are available which may hold adequate training data for a supervised ML approach, *viz.* the GENIA corpus (Ohta et al., 2002) and the PENNBIOIE corpus (Kulick et al., 2004). Whereas determining the sentence boundaries in a text corpus may be seen as a comparatively easy task, deciding on token boundaries is not as straightforward as it might appear, at first sight. In particular, in the biomedical sublanguages such as evidenced in PUBMED abstracts¹ (or in full articles), crucial semantic units, e.g., entity names or even references to biological processes, can be contained within larger string units and, hence, are not simply delimited by white spaces. A closer look at both corpora reveals that only PENNBIOIE addresses the problem of semantically motivated word token boundary annotation at all, whereas GENIA annotates tokens around the same closed set of special characters as used for the English newspaper language (Marcus et al., 1993).

In this paper, we first report on the compilation and annotation of the JULIE Corpus which provides sentence and word token boundary information in a semantically motivated and linguistically plausible way. The JULIE Corpus is composed of documents from a large variety of biomedical subdomains and entity types with critical word token phenomena. We then report on the JULIE Tools, ML-based tools for sentence and token boundary detection. We evaluate our corpus against GENIA and PENNBIOIE and our tools against another

¹PUBMED is the largest bibliographic database for the life sciences and currently contains more than 16 million entries; see <http://www.ncbi.nlm.nih.gov/entrez/>

ML-based tool suite, the OpenNLP Tools.²

2 Background

To train ML-based tools for sentence and/or token boundary detection, high-quality annotated resources are needed. In particular, several ambiguous non-alphanumeric character symbols may denote sentence or word token boundaries (or not) and, furthermore, these symbols may (or may not) be part of names for biomedical entities, such as protein, cell, or organism names etc. In the following, we outline the annotation guidelines considered for the compilation of the JULIE Corpus which we annotated with sentence and token boundary information.

2.1 Sentence Boundary Annotation

For sentence boundary annotation, it is first necessary to determine potential *sentence boundary symbols* (SBS). For biomedical language texts, such as those from the PUBMED literature database, we defined the ‘classical’ sentence boundary symbols (“;”, “!”, “?”, “:”) and also two PUBMED-specific ones (“)”, “}”). In particular, for periods (“.”) and colons (“:”) we encountered many cases where they did not denote an SBS:

- General abbreviations (“e.g.”, “i.e.”, “et al.”, “ref.”, “viz.”, “Dr.”, “vs.”, etc.);
- Numbers (“0.05”, “1.2”, “.4”);
- Entity names, e.g., organism names (“E. coli”, “F. oxysporum”, “f. sp. Lycopersicim”, “P. decumbens”). Classifying these symbols as SBS would break up organism names which are essential for disambiguating protein names and mapping them into their database entry (e.g., UniProt). Obviously, simply stating a rule that marks an SBS after each period if the following word starts with a capital letter would over-split the organism entity “f. sp. Lycopersicim”. Moreover, it would also fail to recognize an SBS in cases where the beginning of the following sentence starts with a lower-case letter, as is the case with many protein names (e.g., “p53”, “tac”, etc.);
- Author and journal names in literature citations which are contained in many PubMed

²<http://opennlp.sourceforge.net>

abstracts (e.g., “Am. J. Physiol.”, “J. Biol. Chem.”, “L. Hoffmann”, “Schindler L.”);

- Other alphanumeric strings, such as EC numbers, chromosome locations, database identifiers (“EC1.7.3.3”, “LEN.PK113-78”);
- Colons followed by enumerations (“Several cytokines interact with each other: IL-2, IL-5, and IL-18.”).

2.2 Token Boundary Annotation

For the annotation of word tokens in biomedical text (or any other domain), it is essential to determine which word tokens denote semantic units (i.e., entities) of interest and thus should be recognized as such (e.g., by named entity recognizers). In biomedical text, there are various symbols which may (or may not) denote word token boundary symbols (TBS), such as “-”, “+”, “/”, “'”, “-”, “%”, “(“, “)”, etc., and thus are heavily ambiguous with respect to their status as token boundary symbols. The most important cases are:

- Parentheses must usually be split from regular words, but, e.g., in chemical terminology they are part of the name, such as in “Ca(2+)” or “(S,S)-Tartate”. The same holds for enumeration list items such as “(1)”, “(2)”, “a)”, “b)”, etc.;
- Plus (“+”) symbols and hyphens may denote relevant semantic information, such as indicating the presentation (+) of an antigen on a cell or the absence (-) thereof: “CD34(+) T-cells”, “CD83+ dendritic cells”, “CD11c(++) B-lymphocytes”, “CD8alpha(-) DCs”. Here, it can be seen that biomedical language expresses complex biological processes by means of a single character symbol. In such cases, of course, these symbols should be tokens on their own;
- Hyphens often concatenate entity names (such as protein names) with other words (“IL-2-specific”,³ “CD28-dependent”, etc.) or even with other entity names, such as cell names, as in “CD43-DC”. Lacking recognition of these entities as word tokens would prevent entity recognizers from detecting them at all;

³Here, only the second hyphen should be split because the first one is actually part of the protein name “IL-2”.

- Similar observations can be made with respect to slashes (“/”) which often separate two (or more) entity references (“IL-2/CD34”, “HA-1/2”, etc.);
- Hyphens (and slashes) may also denote the knock-out status of a certain gene with respect to an organism, such as in “flt3L-/- mice”.

One might argue that such problems can be overcome by simply splitting (i.e., marking a sentence or token boundary symbol) at every potential split symbol (i.e., at every parenthesis, hyphen, period, colon, etc.). This strategy would, however, split a protein name such as “IL-2” into [IL], [-], and [2]. However, modules further up in a typical text mining pipeline (part-of-speech taggers, phrase chunkers, syntactic parsers, etc.) would not be able to perform adequately on such broken segmentation data because their linguistic representations (either rule-based or derived from training data) could not deal at all with such fragments.

3 Corpus and NLP Tools

3.1 JULIE Sentence and Token Corpus

Currently, only one biomedical text corpus, PENNBIOIE, addresses both sentence boundary and tokenization annotation issues at all, although insufficiently. Furthermore, it is limited to two highly specialized biomedical subdomains, viz. the CYP450 enzyme and oncology, focusing on few organism types (human and mouse). No manual token annotation was done on the second well-known annotated biomedical corpus, GENIA, whose domain scope is also rather limited (transcription factors in human blood cells).

Using different sets of MeSH terms⁴, we assembled a text corpus consisting of abstracts from PUBMED which covered a more varied set of biological subdomains, including gene expression and regulation, stem cell biology/transplantation and immunology. We refer to this as the *Subdomain Corpus*. Additionally, again by using the MeSH thesaurus, we also assembled subcorpora with respect to biomedical entity types which are known to pose severe problems for sentence and token boundary detection, viz. organisms, chemi-

⁴<http://www.nlm.nih.gov/mesh/meshhome.html>, MeSH is the U.S. National Library of Medicine’s controlled vocabulary used for indexing PUBMED articles.

Subdomain + Entity Corpus	20,244
GENIA	18,777
PENNBIOIE	23,358
total	62,379

Table 1: Composition of the JULIE Sentence Corpus (number of sentences)

Subdomain Corpus	12,566
PENNBIOIE	23,358
total	35,924

Table 2: Composition of the JULIE Token Corpus (number of sentences)

icals, cell types and cell components. We refer to this as the *Entity Corpus*.

The Subdomain and Entity Corpora for sentence boundary information were automatically preprocessed with the OpenNLP Sentence Splitter trained on the PENNBIOIE corpus (Buyko et al., 2006). Then the annotations were manually inspected and, if necessary, corrected by a computational linguist and a biologist. In addition, we have added the sentence annotations from GENIA and PENNBIOIE. Altogether, the **JULIE Sentence Corpus** contains 62,379 annotated sentences (see Table 1).

A similar procedure was performed for token annotation. Due to the more complex task, only the Subdomain Corpus was annotated with word token boundary information. Furthermore, we corrected the token annotations in the PENNBIOIE corpus when it did not conform to our guidelines. The **JULIE Token Corpus** contains both the annotated Subdomain Corpus and the corrected PENNBIOIE data, summing up to 35,924 sentences (see Table 2).

3.2 ML-Based JULIE Tools

Our tools for sentence splitting and tokenization are based on supervised machine learning, i.e., from a given set of training examples a statistical model is learned. Such a model is then used to predict labels for unseen data.

Both tasks, i.e., sentence splitting and tokenization, can be considered similar to other chunking tasks where a sequence of (linguistic) units needs to be assigned to labels. This may be interpreted as a traditional classification task where all units of a sequence are considered mutually in-

dependent⁵ and their labels are thus predicted separately. For the sentence splitter, such a sequence is the complete input text⁶ which is decomposed into units of character sequences (split at white space positions). The tokenizer, on the other hand, receives as input sequence a sentence; here, the single units are atomic character sequences which may be combined to tokens.

If, however, there is an inherent interdependency between the single units in a sequence and the independence assumption does not hold, this may lead to a suboptimal classification performance. This is the case in many NLP chunking tasks, including POS tagging, shallow parsing, and especially entity recognition. We assume this to be the case with sentence splitting and tokenization as well. Under the presence of such statistical relational data, the application of sequence models, which assign a sequence of labels to the complete input sequence of units, should be favored due to the natural dependency structure.

Conditional Random Fields (Lafferty et al., 2001) are probabilistic sequence models which have recently been shown to be well suited for different NLP tasks (cf. (Settles, 2004) for named entity recognition, (Sha and Pereira, 2003) for parsing). A Conditional Random Field (CRF) can be interpreted as an undirected graphical model which is globally conditioned on the observation sequence \mathbf{o} of length n and trained so that it maximizes the conditional probability of the label sequence $\mathbf{l} = \langle l_1, \dots, l_n \rangle$ and the observation sequence $\mathbf{o} = \langle o_1, \dots, o_n \rangle$. A CRF can be arbitrarily structured; however, in most NLP applications, the underlying graph is a simple linear chain. In the following, we will only focus on this case. In a CRF the conditional probability of an observation sequence given a label sequence is thus defined as:

$$P(\mathbf{l}|\mathbf{o}) = \frac{1}{Z(\mathbf{o})} \exp \left(\sum_{i=1}^n \sum_k \lambda_k f_k(l_{i-1}, l_i, \mathbf{o}, i) \right)$$

where $Z(\mathbf{o})$ is a normalization factor which guarantees that we yield a proper probability. A feature function f_k relates the current observation with a bigram of labels; i.e., it is an indicator function associating an observation feature at position

⁵Formally, this is based on the assumption that all these units are independent and identically-distributed examples.

⁶In case of an abstract, the complete abstract may be considered as one sequence. When handling full texts, one might handle each paragraph separately.

i (e.g., *unit at current position i is “,”*) and a particular state transition (e.g., *state at position $i - 1$ is “N” and state at current position i is “P”*). The feature weights λ_k are estimated by maximizing the respective conditional log-likelihood function on the training data by means of iterative-scaling or gradient-based techniques. The higher the value of λ_k , the more important this feature function is.

As an implementation of CRFs, we employed the machine learning toolkit MALLET (McCallum, 2002). In the following, the sentence and the token splitter, referred to as JULIE Tools⁷, are explained in detail.

As a general rule, we tried to optimize our tools not only in terms of accuracy, but also with respect to the kind of errors made, *viz.* false positive (FP) and false negative (FN) errors. As for the sentence splitter, we prefer FNs over FPs. As many consecutive NLP components work on the sentence level the costs of erroneously splitting a sentence should be considered higher than not splitting. Here, a false positive means a loss of information which might be especially problematic, e.g., when running an entity tagger or a parser on such a dissected sentence. For the tokenizer, we favor FPs over FNs because in many consecutive processing steps tokens are considered as atomic units; however, if the segmentation is too coarse, again information is lost.

3.2.1 JULIE Sentence Boundary Detector

The input text is broken down into a sequence of (observation) units by splitting at all white space positions. For each such unit, the JULIE Sentence Boundary Detector (JSBD) has to decide whether the end of the sentence has been reached or not. Thus, the following piece of text from a PubMed abstract would be split into the following units (with the actual sentence boundary symbol after “T-cells”):

```
... [on] [IL-2-activated] [CD34(+)]
[cytotoxic] [T-cells.] [p3hr-1,] [the]
[Burkitt's] [lymphoma] [cell] [line,]
[was] ...
```

Each unit is then represented by the following set of binary features:

- The unit itself (lexical feature) and its size in characters;

⁷A Java implementation of our tools can be downloaded from our website: <http://www.julielab.de>.

- Sentence boundary symbols (SBS):⁸ whether the unit ends with SBS, whether the unit contains SBS;
- Canonical word form: transformation rule which replaces capital letters by “A”, lowercase letters by “a”, digits by “0”, and all other characters by “-”. Finally identical consecutive letters are collapsed (e.g. IL2→AA0→A0);
- Orthographical features: based on regular expressions (e.g. HasDash, AllCaps, InitalCap, hasParenthesis, ...);
- Abbreviations: whether the unit is contained in the list of known abbreviations, whether the unit conforms to abbreviation classes (“[A-Z].”, “([A-Za-z].)+”, “[a-z].+”);
- Local context: features of neighboring units in window [-1,1] copied to current unit.

JSBD also employs a rule-based post-processing routine to avoid that a sentence is split within opened parentheses or brackets. This is a useful extension, as scientific papers and abstracts often contain complex bibliographic references within parenthetical or bracket-like expressions. Such a reference should be considered as one sentence. However, within these parenthesized sentences there are often many SBSs, especially periods, which could also be considered as sentence boundaries. Preliminary experiments showed that this processing routine, though it does not improve the overall performance, shifts the FP/FN ratio favorably, *i.e.*, false positives are avoided (clearly at the cost of increasing the rate of false negatives).

3.2.2 JULIE Token Boundary Detector

For the tokenizer we have defined a set of critical *token boundary symbols*⁹ (TBS) where possible token boundaries might occur. It should be noted that periods, exclamation and question marks are not included in the TBS set because these are sentence boundary symbols and have been disambiguated by the sentence splitter before. This is in line with the paradigm of running NLP tools in a (sequential) pipeline, where each

⁸Sentence boundary symbols are, e.g., period, question and exclamation mark, etc.

⁹The complete set of token boundary symbols consists of { } , + - () [] ; = / & > < ' .

tool is designated to one NLP task and a (natural) order is implemented for the tools due to inherent input/output dependencies. Thus, tokenization is typically performed after sentence splitting.

At all white space positions and at each TBS we split the sentence into single units, the TBS itself is treated as a separate unit. Our example piece of text from above would thus contain the following units:

```
... [on] [IL] [-] [2] [-] [activated] [CD34]
[ ] [+ ] [ ] [cytotoxic] [T] [-] [cells] SBS
[p3hr] [-] [1] [,] [the] [Burkitt] ['] [s]
[lymphoma] [cell] [line] [,] [was] ...
```

For each such unit, JTBD decides whether this unit constitutes the end of a token or not. A token thus consists of a sequence of n units with the last unit labeled as token end and the other $n - 1$ units as inside a token.

Furthermore, we assign each unit to its so-called *super-unit*. So, we also split the sentence into larger strings at each white space position as we did for sentence splitting. Such a string is considered a super-unit of a unit, if it covers this unit in the sentence. In our example piece of text from above, we would construct the following super-units:

```
... [on] [IL-2-activated] [CD34(+)]
[cytotoxic] [T-cells.] [p3hr-1,] [the]
[Burkitt's] [lymphoma] [cell] [line,]
[was] ...
```

The super-units are used as context information when we construct the features. Each unit is then represented by the following set of binary features:

- The unit itself, and the super-unit itself (lexical feature);
- Whether the unit had a white space to the right in the original sentence;
- Whether the unit is a TBS;
- Features equivalent to the sentence splitter: size, canonical word form, abbreviation class, local context, rich orthographical features of the unit;
- Bracketing information: whether the super-unit contains brackets (*hasOpeningBracketOnly*, *isInBrackets*, *hasClosingBracketOnly*,...);

- Whether the super-unit is an enumeration (“(1)”), whether the super-unit has genitive (“enzyme’s”), whether the super-unit has plural in brackets (“enzyme(s)”);
- Other orthographical features of the super-unit with focus on hyphens, arrows, +/- symbols which are often contained in biomedical texts, and thus structure chemical names.

4 Experimental Results and Discussion

Sentence splitting and tokenization performance is typically evaluated in terms of the accuracy (A), i.e., the number of correct decisions divided by the total number of decisions being made. Here, the total number of decisions equals the number of units. In our evaluation, we focus on two questions, viz. (1) how well are the different corpora suited for training, and, (2) whether the JULIE Tools perform better than another well-known ML-based NLP tool suite, the OpenNLP Tools, whose general applicability to the biomedical domain has already been shown (Buyko et al., 2006). Among other NLP tools, OpenNLP also provides a sentence splitter and a tokenizer, both based on conditional maximum entropy models (Ratnaparkhi, 1998), also known as logistic regression. Compared to the rich feature sets of the BioMed Tools, the OpenNLP counterparts have much fewer features (mainly lexical ones) coupled with a non-sequential learning algorithm.

4.1 Sentence Splitter Evaluation

To address the first question, we trained the JULIE Sentence Boundary Detector on both the GENIA corpus (18,529 sentences with approximately 486,000 word tokens) and the original PENNBIOIE corpus (23,277 sentences with approximately 590,000 word tokens). The models learned from this training material were then evaluated against the Subdomain and Entity Corpora. We did not evaluate against the complete JULIE Sentence Corpus because it comprises both the GENIA and the PENNBIOIE corpus. The results are depicted in Table 3. There are only very small differences in the accuracy of the two models (A=99.58 and A=99.62). In addition, we also trained and cross-validated JSBD on the complete JULIE Sentence Corpus which yields an accuracy of A=99.8. Here, we encounter a small improvement over the last two experiments. Table 3 also shows the performance of OpenNLP’s sentence

tool	training data	evaluation data	A	FP
JSBD	GENIA	Subdomain and Entity Corpus	99.58	30%
JSBD	PENNBIOIE	Subdomain and Entity Corpus	99.62	27%
JSBD	10-fold cross-validation on JULIE Sentence Corpus		99.80	30%
OpenNLP	10-fold cross-validation on JULIE Sentence Corpus		98.70	48%

Table 3: Sentence splitter performance: Evaluated on different tools and different corpora

tool	training data	evaluation data	A	FP
JTBD	GENIA	JULIE Token Corpus	71.50	3%
JTBD	PENNBIOIE	JULIE Token Corpus	95.90	25%
JTBD	10-fold cross-validation on JULIE Token Corpus		96.70	45%
OpenNLP	10-fold cross-validation on JULIE Token Corpus		95.00	47%

Table 4: Tokenizer performance: Evaluated on different tools and different corpora

splitter on the complete JULIE Sentence Corpus (10-fold cross-validation). In this setting, JSBD performed significantly better¹⁰ than OpenNLP’s sentence splitter (A=99.8 vs. A=98.7).

This result can be explained by the rich feature set in our sentence splitter and the fact that we consider sentence splitting as a sequential learning problem. Also, our sentence splitter produces fewer false positives (FP) than OpenNLP’s sentence splitter (FP=30% vs. 48%), which is more favorable for this task.

4.2 Tokenizer Evaluation

We trained the JULIE Token Boundary Detector on GENIA and PENNBIOIE, and evaluated the models on the JULIE Token Corpus to determine how well these corpora are suited for training (see Table 4). As GENIA is only tokenized using newspaper language patterns, it is not so well suited for training; only an accuracy of about A=71.5% is reached. PENNBIOIE is more apt because its word token annotation is more semantically motivated. A 10-fold cross-validation of JTBD on the complete JULIE Token Corpus shows that the performance is thus improved by approximately 1 percentage point. Table 4 also indicates that in comparison with the 10-fold cross-validation of OpenNLP’s tokenizer on the complete JULIE Token Corpus (A=95.0), both JTBD’s machine learning algorithm (sequential learning) and its rich linguistic feature representation (super-units/units) are superior (A=96.7). Looking at the tokenization decisions, the OpenNLP tokenizer runs into

¹⁰On such a level of accuracy, a difference of 1 percentage point is notable because this affects mostly critical cases (organism names, etc.).

particular problems with hyphens which either are not split at all or, if split, in a rather inconsistent way. Thus, an expression like “IL-2-activated” is sometimes tokenized as [IL-2][activated], [IL][activated], or not at all. Similar errors occur with expressions such as “CD34(+)”.

From the above experiments we conclude that for sentence splitting the corpus being used for training is not so critical because sentence annotations are not really controversial within different biomedical subdomains. Tokenization, however, is a much more complex task with respect to the relevant biomedical semantic units to be annotated. Still, performance of tokenization can be significantly improved by employing a tool with a linguistically adequate representation and a rich feature set. In both cases, the extension of an annotation corpus by a set of critical (and rare) subdomain and entity cases, as is done in the JULIE Sentence and Token Corpus, boosts performance.

5 Conclusion

We considered two low-level NLP tasks, *viz.* splitting a text document into its constituent sentences and splitting sentences into their constituent tokens. While these tasks seem to have been sufficiently solved for the general-language newspaper domain, they pose considerable problems for sublanguages in the natural sciences and many engineering domains.

As a solution for this challenge, we propose a sentence splitter and a tokenizer based on Conditional Random Fields, a state-of-the-art sequential machine learning algorithm, that is recently applied to many other NLP tasks. We also argued

for rich(er) feature sets, and built novel corpora for the biomedical domain to train these tools on.

The corpora are basically an extension and a correction of the PENNBIOIE corpus – an extension because we added training material so that this corpus covers a more complete section of the biomedical domain and is not specialized to a subdomain; a correction because we removed some inconsistencies with respect to our annotation guidelines.

Our evaluation experiments run on these corpora indicate that both for the sentence splitting and the tokenization task, a substantial improvement in performance could be achieved. Compared to a maximum entropy approach with poorer feature sets (OpenNLP), these results suggest the superiority of an ML approach based on CRFs and a rich, linguistically-motivated feature set for such NLP preprocessing tasks.

Acknowledgements

This research was funded by the EC's 6th Framework Programme (within the BOOTStrep project under grant FP6-028099), and by the German Ministry of Education and Research (BMBF) within the StemNet project (funding code: 01DS001A to 1C).

References

Ekaterina Buyko, Joachim Wermter, Michael Poprat, and Udo Hahn. 2006. Automatically adapting an NLP core engine to the biology domain. In Hagit Shatkay, Lynette Hirschman, Alfonso Valencia, and Christian Blaschke, editors, *Proceedings of the ISMB 2006 "The Joint Linking Literature, Information and Knowledge for Biology and the 9th Bio-Ontologies Meeting"*. Fortaleza, Brazil, August 2006.

Gregory Grefenstette and Pasi Tapanainen. 1994. What is a word, what is a sentence? Problems of tokenization. In *Proceedings of the 3rd International Conference on Computational Lexicography*, pages 79–87.

Claire Grover, Michael Matthews, and Richard Tobin. 2006. Tools to address the interdependence between tokenisation and standoff annotation. In *Proceedings of the 5th EACL-2006 Workshop on NLP and XML (NLPXML-2006): Multi-Dimensional Markup in Natural Language Processing*, pages 19–26. Trento,

Italy, April 4, 2006. East Stroudsburg, PA: Association for Computational Linguistics (ACL).

Seth Kulick, Ann Bies, Mark Liberman, Mark Mandel, Ryan McDonald, Martha Palmer, Andrew Schein, Lyle Ungar, Scott Winters, and Pete White. 2004. Integrated annotation for biomedical information extraction. In *Proceedings of the HLT-NAACL 2004 Workshop 'Linking Biological Literature, Ontologies and Databases: Tools for Users – BioLink 2004'*, pages 61–68. Boston, MA, USA, May 2004.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML-2001 – Proceedings of the 18th International Conference on Machine Learning*, pages 282–289. Williams College, MA, USA, June 28 - July 1, 2001. San Francisco, CA: Morgan Kaufmann.

Yang Liu, Andreas Stolcke, Elizabeth Shriberg, and Mary Harper. 2005. Using conditional random fields for sentence boundary detection in speech. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 451–458. Association for Computational Linguistics, Ann Arbor, Michigan.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The PENN TREEBANK. *Computational Linguistics*, 19(2):313–330.

Andrew K. McCallum. 2002. MALLET: A machine learning for language toolkit. <http://mallet.cs.umass.edu>.

Grace Ngai and David Yarowsky. 2000. Rule writing or annotation: Cost-efficient resource usage for base noun phrase chunking. In *ACL'00 – Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 117–125. Hong Kong, China, 1-8 August 2000. San Francisco, CA: Morgan Kaufmann.

Tomoko Ohta, Yuka Tateisi, and Jin-Dong Kim. 2002. The GENIA corpus: An annotated research abstract corpus in molecular biology domain. In M. Marcus, editor, *HLT 2002 – Human Language Technology Conference. Proceedings of the 2nd International Conference on Human Language Technology Research*, pages 82–86.

- San Diego, Cal., USA, March 24-27, 2002. San Francisco, CA: Morgan Kaufmann.
- David D. Palmer and Marti A. Hearst. 1997. Adaptive multilingual sentence boundary disambiguation. *Computational Linguistics*, 23(2):241–267.
- Fuchun Peng, Fangfang Feng, and Andrew McCallum. 2004. Chinese segmentation and new word detection using conditional random fields. In *COLING'04: Proceedings of the 20th International conference on Computational Linguistics*, pages 562–568. Association for Computational Linguistics, Morristown, NJ, USA.
- Adwait Ratnaparkhi. 1998. *Maximum Entropy Models for Natural Language Ambiguity Resolution*. Ph.D. thesis, University of Pennsylvania.
- Burr Settles. 2004. Biomedical named entity recognition using conditional random fields and rich feature sets. In Nigel Collier, Patrick Ruch, and Adeline Nazarenko, editors, *JNLPBA – Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications*, pages 107–110. Geneva, Switzerland, August 28-29, 2004.
- Fei Sha and Fernando C. N. Pereira. 2003. Shallow parsing with conditional random fields. In *HLT-NAACL 2003 – Proceedings of the Human Language Technology Conference and the 3rd Conference of the North American Chapter of the Association for Computational Linguistics*, pages 134–141. May 27 - June 1, 2003, Edmonton, Canada. San Francisco, CA: Morgan Kaufmann.