A Preliminary Look into the Use of Named Entity Information for Bioscience Text Tokenization

Robert Arens
Department of Computer Science
University of Iowa
Iowa City, Iowa, USA
robert-arens@uiowa.edu

Abstract

Tokenization in the bioscience domain is often difficult. New terms, technical terminology, and nonstandard orthography, all common in bioscience text, contribute to this difficulty. This paper will introduce the tasks of tokenization, normalization before introducing BAccHANT, a system built for bioscience text normalization. Casting tokenization / normalization as a problem of punctuation classification motivates using machine learning methods in the implementation of this system. The evaluation of BAccHANT's performance included error analysis of the system's performance inside and outside of named entities (NEs) from the GENIA corpus, which led to the creation of a normalization system trained solely on data from inside NEs, BAccHANT-N. Evaluation of this new system indicated that normalization systems trained on data inside NEs perform better than systems trained both inside and outside NEs, motivating a merging of tokenization and named entity tagging processes as opposed to the standard pipelining approach.

1 Introduction

For the purposes of this paper, a token can be defined as the smallest discrete unit of meaning in a document relevant to the task at hand, the smallest entity of information that cannot be further reduced in form and still carry that information. This definition of a token is dependent on both the type of information we wish to extract from a document, and the nature of the document itself; that is, tokenization is task-specific. For example, tokenizing technical reports in order to search them by keyword may require a conservative tokenization scheme; e.g. a document containing the term "2.0-gigahertz processor" would not want to tokenize "2.0" away from "gigahertz" for fear that the document would be missed if the user searched for that exact phrase. However, if the same set of documents was being tokenized to build a database of processor speeds, "2.0" would need to be tokenized away from "gigahertz" in order to store its speed

Tokenization is often straightforward; discovering the words in the sentence, "I saw a cat." is not difficult, as the tokens are bounded by punctuation, including space characters. However, discovering the words in the sentence, "I studied E. coli in a 2.5% solution." presents some problems. The period following “E.” is being used to indicate an acronym instead of a sentence boundary, and must be recognized as such. Even if we were not to concern ourselves with sentence boundaries, deciding that any period simply ends a token, the sentence would again present a problem since we would not want to tokenize the 2 from the 5 in "2.5".

The difficulty in tokenization stems from ambiguous punctuation. In order to tokenize, one must be able to tell with certainty when a piece of punctuation ends a token.

The bioscience domain presents additional difficulties to tokenizing. Bioscience literature contains technical terminology and includes ambiguous punctuation, similar to the E. coli sentence above. The domain is dynamic, with thousands of researchers adding to the literature (the MEDLINE database adds approximately 400,000 new entries consisting of journal

articles and abstracts per year (*MEDLINE Fact Sheet*, 2002)). Bioscience literature contains heterogeneous orthographics; for example, the literature contains the terms "NF-kappaB", "NF-kappa B", and "NF-kappa-B," and while each refers to the same protein, tokenizers using spaces and dashes as breaking criteria will return a different tokenization of each term though one standard tokenization would be preferable.

The problem of nonstandard orthography is of particular importance for document retrieval. Consider the NF-kappaB example above; if a document repository contains documents with different orthographic versions of NF-kappaB, a researcher searching for NF-kappaB would have to search for all possible orthographic variations and would miss documents containing unanticipated orthography. Normalization attempts to solve this problem by removing orthographic variation from tokens, bringing them to one normalized form. For example, if all three versions of NF-kappaB had all spaces and dashes removed, all three would look like "NFkappaB," and a document retrieval system would find all instances in a search. A related strategy, query expansion, attempts to solve the same problem by accounting for as many orthographic variants of the user query as possible, and searching for all of them. Normalization acts as a special case of tokenization by deciding which instances of punctuation break a token, and removing all punctuation that does not break the token in order to bring it to a normalized form.

The remainder of this paper will consider work relevant to tokenization and normalization for the bioscience domain. Casting tokenization, and by extension normalization, as a classification problem motivates the creation of BAccHANT, a machine learning system designed to normalize bioscience text. Evaluation of this system includes an evaluation of the system's performance inside and outside of named entities, and results from this evaluation motivate the creation of a new system, BAccHANT-N, trained solely on date from inside NEs. The improvement in performance of BAccHANT-N over BAccHANT when normalizing inside NE text indicates that named entity information is useful for bioscience text tokenization tasks, motivating future work in systems that perform tokenization and NE tagging concurrently.

2 Related work

As noted in Habert et al. (1998), standard methods for evaluating the quality of tokens produced by tokenization systems do not exist. Though a necessary first step to tasks such as document retrieval, sentence boundary finding, parsing, etc., there exists work involving these tasks that take tokenization for granted (e.g. Chang, Schutze and Altman (2002), Seki and Mostafa (2003)), mention tokenization without detailing the tokenization scheme (e.g. Fukuda et al. (1998)), or indicate use of a tokenization system without mentioning its performance (e.g. Bennet et al. (1999), Yamamoto et al. (2003)). To the author's knowledge, there exists no work analyzing the impact of tokenization performance on bioinformatics tasks.

Tokenization methods for bioinformatics tasks range from simple to complex. Bennet et al. (1999) tokenized for noun phrase extraction, tokenizing based on whitespace, with additional modification to take "specialized nomenclature" into account. Yamamoto et al. (2003) developed a morphological analyzer for protein name tagging which tokenized, part-of-speech tagged, and stemmed documents. Seki and Mostafa (2003) essentially tokenized by dictionary lookup for protein name extraction, using hand-crafted rules and filtering to identify protein name candidates to check against their dictionary.

Relevant work on normalization can be found in the proceedings of the 2003 Text REtrieval Conference (TREC) Genomics track competition. The competition involved two tasks. The first task was for gene or protein X, find all MEDLINE references that focus on the basic biology of the gene/protein from the designated organism. Basic biology includes isolation, structure, genetics and function of genes/proteins in normal and disease states. The second task was to extract GeneRIF statements from records from the MEDLINE biomedical and health abstract repository.

Kayaalp et al. (2003) normalized by converting all letters to lower case, and expanded queries by identifying terms with both alphabetic and numerical characters and searching for hyphenated variants, i.e. JAK2 and JAK-2. de Bruijn and Martin (2003) used morphological query expansion along with a relevance feedback engine. Osborne et al. used a number of query expansion strategies, including appending parenthetical information, acronym expansions, words following hyphens, lower and uppercase versions of terms, etc.

de Brujin and Martin (2003) and Osborne et al. (2003) both indicate that query expansion was beneficial to the performance of their systems. However, no authors gave performance measures for their query expansion methods independent of their final systems. To the author's knowledge, there exists no work analyzing the performance of normalization systems for bioscience literature.

Named entities are "proper names and quantities of interest" (Chinchor (1998)) in a document. Named entity tagging involves discovering and marking these entities in a document, e.g. finding all proteins in a document and labeling them as such. Having biomedical documents tagged with NEs allows for better information extraction, archival, searching, etc. of those documents. The GENIA corpus (Kim et al. (2003)) is a corpus of 2000 MEDLINE abstracts tagged for parts of speech and hand-tagged for NEs. NE tags in the GENIA corpus are based on an ontology, consisting of amino

acids, proteins, organisms and their tissues, cells, and other.

3 Methodology

From a machine learning perspective, one way to look at a tokenization task, including normalization, is as a classification problem. As stated before, the problem of tokenization is that of ambiguous punctuation – one must be able to tell whether or not a piece of punctuation should be included in a token. A document can be tokenized by classifying each piece of punctuation in the document as part of a token or as a token boundary. Removing the pieces of punctuation classified as part of the token will normalize the token. Possible features for classifying punctuation may include the piece of punctuation itself, character or characters to the left/right of the punctuation, type of character[s] to the left/right of the punctuation (i.e. uppercase, lowercase, number, etc.), the length of the sentence or article the term occurs in, the type of sentence or article the term occurs in, etc.

The system presented here, BAccHANT (Bioscience And Health Article Normalizing Tokenizer), was created to normalize MEDLINE text for the TREC Genomics track, as presented earlier. It classifies pieces of punctuation in bioscience text based on the surrounding characters, determining whether the punctuation is a token boundary or needs to be removed for normalization.

The features chosen for BAccHANT were the following: piece of punctuation being classified (Punc), character to the left of the punctuation (CL), type of character to the left of the punctuation (TL), character to the right of the punctuation (CR), type of character to the right of the punctuation (TR), and whether the punctuation should be removed for normalization, or break the token (Class). These features were chosen by the author. Feature selection using information gain ratio indicated that all five should be used.

Feature	Values
Punc	. , - ( ) / ; [ ] : \ { } <space>
CL / CR	<the character itself>
TL / TR	lower, cap, num, space, other
Class	remove, break

Table 1: The features and their possible values.

Values for Punc and CL/CR are self-explanatory.
Values for TL/TR are as follows:
* lower: Character is lowercase
* cap: Character is a capital letter
* num: Character is a number
* space: Character is whitespace (space, tab, etc.)
* other: Character is none of the above
Values for Class are as follows:
* remove: The punctuation should be removed
* break: The punctuation should break the token

The 'remove' class is of chief importance for the normalization task, since classifying a piece of punctuation as 'remove' means the punctuation will be removed for normalization.

Sample feature vectors:
* "NF-kappaB" == ['F', -, 'k', cap, lower, remove]
* "T cells" == ['T', , 'c',cap, lower, remove]
* "alpha gene" == ['a', , 'g',lower, lower, break]
* "yATF-binding" == ['F', -, 'b', cap, lower, break]

The training / testing set for BAccHANT was constructed from 67 MEDLINE abstracts, hand tokenized by the author using the tokenization scheme presented in the appendix. A domain expert[1] was available for determining difficult tokenizations. The 67 abstracts yielded 17253 pieces of punctuation. Distributions follow. The feature vectors created from the set were used to create a decision tree, implemented using the Weka tool set (Witten and Frank). The tree used reduced error pruning to increase accuracy.

Punctuation			
Type	**Total**	**remove**	**break**
<space>	14476	463	14013
-	1103	737	366
.	637	12	625
,	577	6	571
(	186	8	178
)	186	7	179
/	45	7	38
:	18	0	18
[	9	6	2
]	9	7	2
;	7	2	5
Totals	**17253**	**1255**	**15998**

Table 2: Punctuation distribution of the MEDLINE train/test set

4 Evaluation

The baseline used for evaluation was to simply break on every instance of punctuation; that is, assume no punctuation needs to be removed. This achieves an accuracy of 92.73%, where accuracy is the percentage of correctly classified punctuation. This baseline was chosen for its high accuracy; however, as it is a simple majority class baseline which always predicts 'break', giving it a precision score of 1, a recall score of 0, and an f-measure of 0 for the 'remove' class.

BAccHANT was trained and tested using 10-fold cross-validation. It achieved an accuracy of 96.60%, which was a statistically significant improvement over the baseline (all significance testing was done using a two-tailed t-test with a p-value of 0.05). More detailed results follow.

[1] Dr. Vladimir Leontiev, University of Iowa, Department of Anatomy and Cell Biology

	Class	
	remove	**break**
Precision	0.832	0.974
Recall	0.668	0.989
F-Measure	0.741	0.982

Table 3: Precision, recall, and f-measure

The 'break' classification reached high precision and recall. This is unsurprising as 96.7% of all <space> punctuation classified as 'break', and <space> punctuation made up 83.9% of all punctuation. Commas and periods were similarly easy to classify as 'break'. Of more interest is the 'remove' classification, as this class indicates punctuation to be normalized. The recall was not as good as was hoped, with BAccHANT discovering roughly 2 out of every 3 instances present, though it correctly classified roughly 5 out of 6 instances it found

We suspected that punctuation was being used differently inside of named entities vs. outside of NEs. To investigate this suspicion, we tested BAccHANT on NE data from the GENIA corpus. The testing set created from GENIA consisted wholly of character data from inside NEs. The set contained 5798 instances of punctuation. Punctuation distribution for the GENIA corpus test set follows.

Punctuation			
Type	**Total**	**remove**	**break**
<space>	4849	157	4692
-	304	192	112
.	237	4	233
,	187	2	185
(	62	3	59
)	62	3	59
/	14	4	10
:	2	0	2
[	0	0	0
]	0	0	0
;	0	0	0
Totals	**5798**	**365**	**5433**

Table 4: Punctuation distribution in the GENIA corpus test set

The accuracy of BAccHANT on this test set was 90%. More detailed results for the 'remove' class follow.

	BAccHANT performance	
Test Set	**All text**	**GENIA corpus**
Accuracy	0.966	0.900
Precision	0.832	0.546
Recall	0.688	0.453
F-Measure	0.741	0.500

Table 5: Accuracy, precision, recall, and F-measure for BAccHANT tested on all text vs. inside NEs. Precision, recall and f-measure are given for the 'remove' class

Further testing revealed that accuracy outside NEs was near 99%. The statistically significant degradation in performance of BAccHANT inside NEs vs. performance both inside and outside NEs indicates that data inside named entities is more difficult to normalize than data outside named entities.

These results seem to indicate that a normalization system trained solely on data inside NEs could perform better than a system trained on both named and non-named data when normalizing NEs. A new normalization system trained on NE data, BAccHANT-N, was built to test this.

The new system was trained and tested using the GENIA corpus test set. BAccHANT-N was created similarly to BAccHANT, with identical features, and implemented as a decision tree using reduced error pruning. It was trained and tested using 10-fold cross-validation and achieved an accuracy of 96.5%. More detailed results follow.

	Class	
	remove	**break**
Precision	0.833	0.980
Recall	0.789	0.985
F-Measure	0.811	0.983

Table 6: Precision, recall, and F-measure for BAccHANT-N tested on named entity data.

Below is a results summary table, giving accuracy for both classes, and precision, recall, and f-measure for the 'remove' class across all systems presented. BAccHANT-N showed statistically significant improvement over BAccHANT when normalizing named entity data. These results show that a system trained on data inside NEs shows improvement in performance over a system trained on data from inside and outside NEs.

	Baseline		*BAccHANT*			
Training set			**All Text**		**Named Entities**	
Test set	All	NE	All	NE	All	NE
Accuracy	0.927	0.914	0.966	0.900		0.965
Precision	1	1	0.832	0.546		0.833
Recall	0	0	0.688	0.453		0.789
F-Measure	0	0	0.741	0.500		0.811

Table 7: Results summary across all systems. Precision, recall, and f-measure are given for the 'remove' class.

5 Future Work

Currently, BAccHANT looks only at one character to either side of the piece of punctuation to be classified.

By expanding the number of characters examined from one to a certain number of characters (a window), accuracy should increase. Since BAccHANT decision tree learns based on context, greater context may allow for better learning, and a window of characters will expand context.

Also, a window of characters will introduce new features to learn from. Since a decision tree's features determine how it learns from context, adding better features to the decision tree may help the tree learn better. Examples of new features include:

* Mixed case - does the window include both uppercase and lowercase characters?

* Mixed type - does the window include a mix of letters, numbers, and other character types?

* Boundary size - is there a definite token boundary within the character window, and if so, how far into the window is the boundary?

Error analysis of BAccHANT on named entity tagged data led to the creation of a normalization system trained on data from inside NEs which performed better than BAccHANT, and hence would be a better choice for normalizing inside NEs. However, this normalizer would necessarily need to be run on named entity tagged data, as it has not been trained to deal with text outside of NEs. To accomplish this, a system to simultaneously tag named entities and normalize at the same time would be desirable. This could be accomplished via hierarchical hidden Markov models (Fine et. al., 1998). A system of this type involves "tiering" hidden Markov models within each other. This model could be used to statistically compute the most likely name for a section of text, and then normalize appropriately in one pass. As hidden Markov models have been used both for name-finding (Bikel et al. (1997)) and tokenization (Cutting et al. (1992)), this seems to be a promising research possibility.

6 Conclusion

This paper has introduced a system to normalize bioscience and health articles based on learning features surrounding punctuation which may need to be removed for normalization. The system performed significantly better than the baseline system.

By analyzing the system's performance on named entity data from the GENIA corpus, it was discovered that named entities seemed to be more difficult to normalize than surrounding non-named text. This finding led to the creation of another normalization system trained on named entity data, which showed significant improvement over the first system when tested on named entities. This improvement seems to indicate that a system which would compute named entities in parallel with normalization would be useful.

References

Nuala A. Bennet, Qin He, Kevin Powell, and Bruce R. Schatz. 1999. Extracting noun phrases for all of MEDLINE. In *Proceedings of the AMIA Symposium,* 671-65

Daniel M. Bikel, Scott Miller, Richard Schwartz and Ralph Weischedel. 1997. Nymble: a High-Performance Learning Name-finder. In *Proceedings of the Conference on Applied Natural Language Processing, 1997.*

Stephen Blott, Cathal Gurrin, Gareth J. F. Jones, Alan F. Smeaton, and Thomas Sodring. 2003. On the Use of MeSH Headings to Improve Retrieval Effectiveness. In *Proceedings of The 12th Text Retrieval Conference, Gaithersburg, Md, November 2003.*

Eric W. Brown, Andrew Dolbey, Lawrence Hunter. 2003. IBM Research and the University of Colorado TREC 2003 Genomics Track. In *Proceedings of The 12th Text Retrieval Conference, Gaithersburg, Md, November 2003.*

Berry de Brujin, and Joel Martin. 2003. Finding Gene Function Using LitMiner. In *Proceedings of The 12th Text Retrieval Conference, Gaithersburg, Md, November 2003.*

Jeffery T. Chang, Hinrich Scuhtze, and Russ B. Altman. 2002. Creating an Online Dictionary of Abbreviations from MEDLINE. *Journal of American Medical Informatics Association*, 9(6): 612-620.

Nancy A. Chinchor. 1998. Overview of MUC-7/MET-2. In *Proceedings of the Seventh Message Understanding Conference (MUC-7).*

Doug Cutting, Julian Kupiec, Jan Pedersen, and Penelope Sibun. 1992. A practical part-of-speech tagger. In *Proceedings of the Third Conference on Applied Natural Language Processing*. 133-140

Shai Fine, Yoram Singer, and Naftali Tishby. 1998. The hierarchical hidden Markov model: Analysis and applications. *Machine Learning*, 32(1):41-62

Ken-ichiro Fukuda, Tatsuhiko Tsunoda, Ayuchi Tamura, and Toshihisa Takagi. 1998. Toward Information Extraction: Identifying Protein Names from Biological Papers. In *Proceedings of the Pacific Symposium on Biocomputing '98 (PSB'98).*

B. Habert, G. Adda, M. Adda-Decker, P. Boula de Mareuil, S. Ferrari, O. Ferret, G. Illouz, and P. Paroubek. 1998. Towards Tokenization Evaluation. In *Proceedings of LREC-98,* 427-431.

William R. Hersh and Ravi T. Bhupatiraju. 2003. TREC Genomics Track Overview. In *Proceedings of The 12th Text Retrieval Conference, Gaithersburg, Md, November 2003.*

Lynette Hirschman, Alexander A. Morgan, and Alexander S. Yeh. 2002. Rutabaga by any other name: extracting biological names. *Journal of Biomedical Informatics*. 35(4): 247-259.

MEDLINE Fact Sheet. (2002). Retrieved November 2, 2003 from http://www.nlm.nih.gov/pubs/factsheets/medline.html

Mehmet Kayaalp, Alan R. Aronson, Susanne M. Humphrey, Nicholas C. Ide, Lorraine K. Tanabe, Lawrence H. Smith, Dina Demner, Russell R. Loane, James G. Mork, and Olivier Bodenreidera. 2003. Methods for accurate retrieval of MEDLINE citations in functional genomics. In *Proceedings of The 12th Text Retrieval Conference, Gaithersburg, Md, November 2003.*

Jun'ichi Kazama, Takaki Makino, Yoshihiro Ohta, Jun'ichi Tsujii. 2002. Tuning Support Vector Machines for Biomedical Named Entity Recognition. In *Proceedings of the Workshop on Natural Language Processing in the Biomedical Domain.* 1-8.

Jin-Dong Kim, Tomoko Ohta, Yuka Tateisi and Jun-ichi Tsujii. 2003. GENIA corpus – a semantically annotated corpus for bio-textmining. *Bioinformatics,* 19(1):180-182.

Andrei Mikheev. 2003. Text Segmentation. In R. Mitkov (Ed.), *The Oxford Handbook of Computational Linguistics* (pp. 201-218). New York: Oxford University Press, Inc.

Miles Osborne, Jeffrey Chang, Mark Cumiskey, Nipun Mehra, Veronica Rotemberg, Gail Sinclair, Matthew Smillie, Russ B. Altman, and Bonnie Webber. 2003. Edinburgh-Stanford TREC 2003 Genomics Track: Notebook Paper. In *Proceedings of The 12th Text Retrieval Conference, Gaithersburg, Md, November 2003.*

David D. Palmer. 1994. Satz - An Adaptive Sentence Segmentation System. M.S. Thesis and UC-Berkeley Technical Report UCB/CSD 94-846.

Kazuhiro Seki and Javed Mostafa. 2003. An Approach to Protein Name Extraction using Heuristics and a Dictionary. Retrieved November 11, 2003, from lair.indiana.edu/research/capris/papers/lair03-04.pdf

Lorraine Tanabe and W. John Wilbur. 2002. Tagging Gene and Protein Names in Full Text Articles. In *Proceedings of the Workshop on Natural Language Processing in the Biomedical Domain.* 9-13.

Ian H. Witten and Eibe Frank. 1999. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations.* San Francisco: Morgan Kaufman Publishers.

Kaoru Yamamoto, Taku Kudo, Akihiko Konagaya, and Yuji Matsumoto. 2003. Protein Name Tagging for Biomedical Annotation in Text. In *Proceedings of the ACL 2003 Workshop on Natural Language Processing in Biomedicine,* pp. 65-72.

Appendix - Hand tokenizing MEDLINE abstracts for normalization

The goal of this tokenization scheme is to process plain-text MEDLINE abstracts into a tokenized gold standard. The format will be one token per line, with breaking punctuation occupying a line by itself.

The rule of thumb for tokenizing in this fashion is, include only punctuation critical for the unique naming of proteins, genes, compounds, etc. found in bioscience literature. Else, the punctuation should be broken on.

Expanded forms of acronyms present an ambiguity problem for tokenization. While we want to keep the acronym “NF-kappa B” as one token, its expanded form “nuclear factor-kappa beta” should be tokenized on all punctuation. While the heterogeneous orthography of “NF-kappa B” must be taken into account since “NF-kappaB” and “NF-kappa-B” both appear in the literature, the literature does not contain instances of “nuclearfactor-kappa beta” or “kappabeta”.

Dashes represent the greatest punctuation ambiguity in the literature, with two out of three instances being removed for normalization. In particular, break if:

* there is a prefix before the dash, as in “anti-DNA” or “non-IL”.

* the dash indicates a number range, as in “1-3 hours”.

* the token candidate following the dash is some kind of modifying noun, gerund or adjective as in “REL-binding” or “Duffy-negative”.

* there are multiple dashes stringing a number of tokens together, as in “neck-spine-torso axis”.

* the dash indicates a negative number.

The gold standards used for training and testing are available from the author by request, or by download at:

http://que.info-science.uiowa.edu/~bob/name-gold (data from inside named entities)

http://que.info-science.uiowa.edu/~bob/all-gold (data from inside and outside named entities)