Parenthetically Speaking: Classifying the Contents of Parentheses for Text Mining

K. Bretonnel Cohen, PhD[1,2], Thomas Christiansen, MS[1],
Lawrence E. Hunter, PhD[1]
[1]Computational Bioscience Program, University of Colorado School of Medicine, Aurora, CO; [2]Department of Linguistics, University of Colorado at Boulder, Boulder, CO

Abstract

The contents of parentheses in biomedical text have many potential uses in text mining applications. However, making use of them requires the ability to determine what class of contents they are. A system that automatically classifies parenthesized text into one of 20 categories is presented and evaluated here. It performs at a micro-averaged accuracy of 68% and a macro-averaged accuracy of 60% on an annotated corpus. The application is available as a Java class and as a Perl module.

Introduction

One of the major upcoming changes to biomedical text mining is a new focus on processing the full text of journal articles. Although systems that deal with full text have been built in the past[1,2,3,4,5], the majority of work in biomedical, especially genomic, natural language processing has dealt with abstracts. However, the creation of PubMedCentral and public access requirements instituted by the National Institutes of Health are poised to make a flood of full-text journal articles available to the text mining community.

One of the identifying features of full-text journal articles is the presence of parenthesized material. One study examined structural and content differences between abstracts and the corresponding article bodies and found that abstracts contained only a small amount of parenthesized text and that parentheses were only used for a limited range of functions[6]. In contrast, they found that article bodies contain many more instances of parenthesized text, and that parentheses are used for a wider range of purposes in article bodies than in abstracts. In fact, they found 17,063 instances of parenthesized text in just the 97 article bodies in the CRAFT corpus[6,7].

Various researchers have pointed out problems related to the presence of parenthesized text. One study noted that it caused problems for parsers, and deleted it entirely[8]. Another study found that it caused errors in hedge scope assignment[9]. Early work noted correctly that parenthesized text in biomedical documents is often confusing to patients and laypeople, but may be useful to biomedical scientists[10]. For example, one popular algorithm uses it for abbreviation definition[11]. Another system used parenthesized text for grounding references to genes to specific entries in the Entrez Gene database[12]. Other researchers have noted that sentences containing parenthesized citations can be put to a number of uses, including establishing rhetorical relations between papers, synonym identification, and locating information for model organism database curation[13].

All of these researchers worked on isolated uses of single types of parenthesized data. We have identified a considerably larger set of use cases for a wide variety of types of parenthesized data. However, to anticipate the remainder of this paper, they all require the ability to first identify what type of data is in the parentheses. We list some use cases in Table 1 (next page) and expand on them in the *Discussion* section. Some of the categories may seem trivially simple to classify. However, that turns out not to be the case. We found five ways to write just the category of statistical test values alone, and had to write the equivalent of a recursive descent parser with a BNF-style grammar to recognize figure references with subfigures—the variability in how authors express these categories is far higher than might be expected. The opposite problem, of ambiguity, exists as well. For example, text strings like +/- are often used to indicate the genotype of experimental subjects, but they may also be used to indicate which panel of a figure to examine.

Methods and Materials

The implementation consists of two decoupled classes (in the technical sense—it is available as Java classes or as Perl modules). The first is optional. Its function is to extract parenthesized material from text. Its main advantages are that it can deal intelligently with unbalanced parentheses (of which we found 47 instances in the 97 documents of the CRAFT corpus[6,7]) and with embedded parentheses (of which we found 76 instances in the CRAFT corpus). The second class actually classifies the text that has been extracted from parentheses.

The classifier itself consists of a number of regular expressions. Many of them are surprisingly complex, with multiple branching conditions. The regular expressions are additionally notable in that they will accommodate a number of aspects of Unicode that would cause non-Unicode-aware regular expressions to make false negative errors in classification (i.e., they would not classify material incorrectly, but rather would fail to classify it at all). This turns out to be an unexpectedly important aspect and advantage of the classifier, and we return to it in the *Discussion* section.

If multiple categories occur within the same set of parentheses, then the classifier returns multiple categories, along with the character offset for each one. For example, given an input like *(0.79 ± 0.05, mean ± SEM compared with 0.76 ± 0.02, p > 0.7, Figure 6C)*, the system returns four labels— *Data, Parenthetical statement, P-value, Table/Figure*—with the character offset for each instance.

Category	Use case
Gene symbol or gene abbreviation	Gene normalization, coreference resolution
Citation	Summaries, high-value sentences, bibliometrics
Data value	Information extraction
P-value	Link weighting, meta-analysis
Figure/table pointer	Strong indicator of good evidence
List element	Mapping sub-figures to text
Singular/plural	Distinguish from other categories
Part of gene name	Gene normalization
Parenthetical statement	Potentially ignorable, or information extraction target

Table 1. Use cases for various types of parenthesized data. These are expanded on in the *Discussion* section.

Evaluation methodology

We evaluated the classifier based on an annotated corpus. An annotator with an advanced degree in the biomedical sciences manually examined 42 articles related to mouse genetics, found every example of parenthesized text, and marked it with the category to which it belonged. The resulting data set contains 7,820 annotations. There was no double annotation, so we do not report inter-annotator agreement values.

When multiple categories appeared within the same set of parentheses, they were all marked separately (see above for a relevant example).

As would be expected, the resulting corpus displayed an imbalance in the categories. Table 2 shows the distribution of categories in the corpus, arranged in descending percentage of instances. We take this imbalance into account in reporting our results, giving both micro- and macro-averaged performance.

Category	Percentage	Count
Parenthetical statement	22.98%	1,797
Figure/Table	22.46%	1,756
Other	19.80%	1548
Citation	8.57%	670
Abbreviation or acronym	6.16%	482
Gene symbol or gene abbreviation	3.96%	310
Data value	3.90%	305
P-value	2.03%	159
NCBI reference	2.01%	157
Descriptive statistics	1.99%	156
Genotype or allele	1.83%	143
Nucleotide sequence	1.25%	98
Statistical test value	0.74%	58
List element	0.73%	57
Appositive	0.69%	54
Part of gene name	0.36%	28
Units	0.24%	19
Singular/plural	0.19%	15
Definition	0.05%	4
Typographical error	0.05%	4
Total annotations	100%	7,820

Table 2. Distribution of categories in the annotated corpus.

Results

We calculate accuracy and present results for each category; micro-averaged results for the entire data set (i.e., averaged over all instances); and macro-averaged results (i.e., averaged over all categories). Table 3 gives the results, ordered alphabetically by category.

Category	Percent correct	Count	
Abbreviation or acronym	67.43%	325/482	
Appositive	0.00%	0/54	
Citation	96.87%	649/670	
Data value	72.46%	221/305	
Definition	0.00%	0/4	
Descriptive statistics	71.79%	112/156	
Figure/Table pointer	66.46%	1,167/1,756	
Gene symbol or gene abbreviation	35.16%	109/310	
Genotype or allele	41.96%	60/143	
List element	0.00%	0/57	
NCBI reference	67.52%	106/157	
Nucleotide sequence	71.43%	70/98	
Other	55.49%	859/1548	
P-value	97.48%	155/159	
Parenthetical statement	78.13%	1404/1797	
Part of gene name	0.00%	0/28	
Singular/plural	100.00%	15/15	

Statistical test value	91.38%	53/58	
Typo	100.00%	4/4	
Units	84.21%	16/19	
Total (micro-averaged)	68.09%	5325/7820	
Total (macro-averaged)	59.89%	N/A	

Table 3. Accuracy for the categories marked in the corpus. The micro-averaged total is the total performance for all 7,820 instances. The macro-averaged total is the average of the accuracies of all categories.

Error analysis

The main contributor to errors in general and to the micro-averaged accuracy in particular is the *Other* category. The problem is the difficulty of differentiating between *Other* and *Parenthetical statement.* Impressionistically, we suspect that the annotation guidelines for these categories need to be sharpened. Ultimately, it may not be important to make this distinction, since there is no use case that requires differentiating between them.

A number of categories with small numbers of members and low performance had negligible effect on the micro-average but a large effect on the macro-average.

Very short strings are difficult to categorize without additional context. For example, the single letter *A* might refer to a figure panel, a list element, or a nucleotide. We discuss our plans to utilize context in the *Future Work* section.

Discussion

This paper has presented an application that parses out data from parentheses and classifies it into a number of classes, most with a defined use case in text mining. We expand here on the use cases sketched briefly in Table 1.

- Gene symbol or gene abbreviation: These categories are useful for gene normalization systems (systems that attempt to map a mention of a gene in text to a specific database entry, e.g. an Entrez Gene ID) and for coreference resolution systems. In the case of gene normalization systems, mapping gene symbols to full gene names is often a crucial step in the algorithm. (We note that gene mention systems may be a better approach to this problem than our regular expressions.) In the case of coreference resolution, mapping gene symbols to gene names is important for finding coreferring noun phrases in text; in fact, it has been hypothesized that in the case of multi-document summarization, the problem of coreference resolution for genes may be reducible to the gene normalization problem.
- Citations: A number of uses have been identified for sentences that contain citations. In the BioCreative shared tasks and in other work[13], it has been shown that sentences that contain citations are often likely to contain information that should be a target for information extraction systems. Citations have also been shown to indicate that a sentence is likely to provide strong evidence for assertions of interest. In building summarization systems, sentences containing citations can indicate information that should be included in the summary, particularly for multi-document summarization systems. Finally, such sentences are a key input to bibliometric studies.
- Data values: Data values constitute a target for some information extraction systems.
- P-values: P-values have several uses. In systems that build networks based on literature analysis[14,15], it might be desirable to weight links in the network by some measure of confidence, which could include the P-value attached to a link. For meta-analyses, it is essential to be able to extract P-values. Finally, some databases include the P-values attached to assertions in the database, e.g. PharmGKB (Y. Garten, personal communication), so there are applications to database curation.

- Figure/Table pointers: Experience from the BioCreative shared tasks suggests that sentences that contain pointers to figures or tables are strong evidence for assertions and may characterize the best evidential sentence in an article. It has also been shown that it is useful to be able to map panels of figures to related sentences in abstracts[16].
- Parenthetical statements: The primary point of interest about parenthetical statements is that by definition, they are incidental to the assertion being made by a sentence, and therefore they can be ignored without harm. This is important for a number of reasons, including knowing when it is not a problem to use the technique described above[8] for improving parser performance by deleting parenthesized text entirely. There is an alternative point of view, which is that parenthetical statements may themselves contain information that should be a target for information extraction; in this case, being able to recognize parenthesis contents as a parenthetical statement informs us what exactly the scope of the text making the assertion should be.
- List elements: List elements are useful for mapping sub-figures to text.
- Singular/plural: The main use for this category is to distinguish the parenthesized *s* from other categories, such as list elements.
- Part of gene name: Parts of gene names are useful for the gene normalization task.

Beyond the individual use cases described above, there are more general reasons why this multi-category system was worth building, versus a proliferation of code written by individual system developers to classify one-off categories of their interest. The following facts suggest that it was worth building and is a valuable tool to have:

- Parenthesized data is pervasive in full-text biomedical journal articles. The work presented here allows for a unified treatment of all such data, and for replicability of work involving any parenthesized data type.
- The system allows for proper handling of Unicode characters.

This last point is so important that we expand on it here. PubMedCentral documents turn out to contain not just a large amount, but an astonishing diversity, of Unicode. At first blush, this might not seem like an issue, since Unicode is handled natively by Java, Perl, and Python. For example, Java has Unicode support built into the language. The type 'char' denotes a Unicode character, and the 'java.lang.String' class denotes a string built up from Unicode characters. However, it turns out that the Java regular expression language is buggy with respect to Unicode. This has repercussions specifically for handling parenthesized data, since a number of characters that are common in parenthesized text in scientific writing are outside of the ASCII character set but in Unicode, e.g. *±, μ, α, β, χ, °, Δ, ≤,* and *≥*. The character ± alone is the third-ranked Unicode character in PubMedCentral Open Access documents as of December 2010, constituting 7.03% of all Unicode characters in those documents. (The second-highest-ranking character is a non-breaking space, which breaks Java's ability to recognize word boundaries in regular expressions.)

Two of the categories that we attempted to distinguish, abbreviations and gene symbols, are difficult to distinguish with regular expressions. Sophisticated system developers might want to combine the application described here with an application specifically designed to detect abbreviation/definition pairs[11] and with one of the many currently available gene mention systems, and use the application described here for the other eighteen categories.

We currently hypothesize that the major contributor to our errors is a combination of a less than optimal set of categories, and a need for better definitions of the categories that we use. Our work was originally driven by a specific set of use cases, but our annotations reflect a larger set of categories. It is likely that some of these should be lumped together. (This would clearly improve our performance—for example, just merging the *Other* and *Parenthetical* categories, which have no clear use case to distinguish them, raises the micro-average from 68% to 76%.) For example, it is not clear from the annotations that there is a meaningful distinction between appositives and definitions. We are currently pursuing this issue.

Future work

The current version of the system examines only the content of the parentheses, ignoring context. This causes a number of problems, including for instance our inability to differentiate between list elements and references to panels of figures. In future revisions, we will allow the program to optionally make use of context if the user wishes to provide it.

Conclusion

This paper presents an application for parsing out and classifying the contents of parentheses, as well as a set of use cases that demonstrates the utility of such an application. The application is available at bionlp.sourceforge.net, and the corpus will be made freely publicly available on completion.

Acknowledgements

Kristina Williams refined the categories and did the corpus annotation. Michael Bada assisted with the set-up of the corpus annotation tool. William A. Baumgartner Jr. and Chris Roeder helped clarify our thinking about Unicode.

References

1. Corney DP, Buxton BF, Langdon WB, Jones DT. BioRAT: extracting biological information from full-length papers. Bioinformatics 2004; 20(17):3206-3213.
2. Friedman C, Kra P, Yu H, Krauthammer M, Rzhetsky A. GENIES: a natural-language processing system for the extraction of molecular pathways from journal articles. Bioinformatics 2001, 17(Suppl. 1):S74-S82.
3. Lin J. Is searching full text more effective than searching abstracts? BMC Bioinformatics 2009, 10(46).
4. McIntosh T, Curran JR. Challenges for automatically extracting molecular interactions from full-text articles. BMC Bioinformatics 2009, 10(311).
5. Agarwal S, Yu H. Automatically classifying sentences in full-text biomedical articles into Introduction, Methods, Results and Discussion. Bioinformatics 2009 25(23):3174-3180.
6. Cohen KB, Johnson HL, Verspoor K, Roeder C, Hunter LE. The structural and content aspects of abstracts versus bodies of full text journal articles are different. BMC Bioinformatics 2010 11:492.
7. Verspoor K, Cohen KB, Hunter LE. The textual characteristics of traditional and Open Access scientific journals are similar. BMC Bioinformatics 2009 10:183.
8. Jang H, Lim J, Lim J-H, Park S-J, Lee K-C, Park S-H. Finding the evidence for protein-protein interactions from PubMed abstracts. Bioinformatics 2006 22(14):e220-e226.
9. Morante R, Daelemans W. Learning the scope of hedge cues in biomedical texts. Proceedings of the BioNLP 2009 workshop 2009, 28-36.
10. Elhadad, N. User-sensitive text summarization: application to the medical domain. 2006 Columbia University PhD thesis.
11. Schwartz AS, Hearst MA. A simple algorithm for identifying abbreviation definitions in biomedical text. Pacific Symposium on Biocomputing 2003.
12. Baumgartner Jr. WA, Lu Z, Johnson HL, Caporaso JG, Paquette J, Lindemann A, White EK, Medvedeva O, Cohen KB, Hunter LE. Concept recognition for extracting protein interaction relations from biomedical text. Genome Biology 2008 9(Suppl. 2).
13. Nakov PI, Schwartz AS, Hearst MA. Citances: citation sentences for semantic analysis of bioscience text. Proceedings of the SIGIR'04 workshop on search and discovery in bioinformatics 2004.
14. Leach SM, Tipney H, Feng W, Baumgartner Jr. WA, Kasliwal P, Schuyler RP, Williams T, Spritz RA, Hunter LE. Biomedical discovery acceleration, with applications to craniofacial development. PLoS Computational Biology 2009.
15. Chen H, Sharp BM. Content-rich biological network constructed by mining PubMed abstracts. BMC Bioinformatics 2004 5(147).
16. Yu H, Lee M. BioEx: a novel user-interface that accesses images from abstract sentences. Proceedings of HLT-NAACL 2006, 189-192.