BIOINFORMATICS

Vol. 00 no. 00 2007
Pages 1–4

From gene names to actual genes

Ioannis Korkontzelos [a]*, Andreas Vlachos [b], Ian Lewin [b]

[a]Department of Computer Science, University of York, York, YO10 5DD, UK,
[b]Computer Laboratory, University of Cambridge, Cambridge, CB3 0FD, UK

ABSTRACT

A common task in biomedical text mining is the recognition of gene names. In many applications though, it is important to know whether a gene name refers to the actual gene or to an entity related to it. This paper presents a trainable system to perform this task. It combines syntactic parsing with SVMs and achieves 78.63% accuracy. The training data used were generated automatically by a simple rule-based tagger. Such an approach can be useful to other fields which exhibit similar ambiguity in the way names are used to refer to entities.

1 INTRODUCTION

There has been a substantial amount of work in recent years on biomedical named entity recognition (Morgan *et al.*, 2004; Blaschke *et al.*, 2004; Kim *et al.*, 2004). It has been observed that gene names (*gn*) are used to name proteins as well as other types of entities. This ambiguity has two consequences. Firstly, it results in inconsistencies in annotation of NER corpora, which renders the evaluation of the various methods problematic (Dingare *et al.*, 2005). Secondly, when a *gn* is recognised in text, it might be important certain to know that it refers to the actual gene, e.g. when we want to identify coreferential chains.

In order to deal with the annotation inconsistencies, Vlachos and Gasperin, 2006 presented guidelines to annotate biomedical text with not only *gns*, but also with information on whether the *gns* refer to the genes. In this paper, we describe a bootstrapping approach based on syntactic parsing and SVMs. Instead of annotating training material manually, we created it automatically using freely available material from FlyBase. Such an approach could be of interest to other fields in which the same name can be used to refer to more than one type of entity and there is lack of annotated material.

2 TASK AND TEST CORPUS DESCRIPTION

The task we deal with is to determine whether a noun phrase (NP) containing one or more *gns* (*gn*) refers to the actual gene(s) or to other biomedical entities. We label these classes as *gene-mention* (*gm*) and *other-mention* (*om*) respectively. For brevity in the remainder of the paper we will be referring to the NPs containing one or more *gns* as *mentions*. In the following sentence, given that "hth" is a *gn*, the task is to classify "the hth gene" as *gm* and "the hth locus" as *om*:

... in <om> the <gn> hth </gn> locus </om> and cloned <gm> the <gn> hth </gn> gene </gm> ...

While the nouns "gene" and "locus" in the respective NPs are indicative, there are many cases where the NP consists of a *gn* only. In such cases, we need to take the context into account.

We used the manually annotated dataset of Vlachos and Gasperin, 2006 as test set. It contains 182 *gms* and 389 *oms*[1].

3 TRAINING DATA CREATION

Annotated material is always the bottleneck when applying machine learning techniques to new tasks and domains. Substantial human effort is required so as to create a reasonably sized and consistently annotated training corpus. In addition, training material created for a task in a particular domain is unlikely to be very useful even in a slightly different domain (Vlachos and Gasperin, 2006). For these reasons, we created training data automatically. The steps performed in this process are described in the following subsections. Briefly, we (a) gathered abstracts and tagged the *gns* in them, (b) parsed the text and extracted the NPs containing *gns* and (c) developed a simple rule-based tagger to classify the NPs.

3.1 Data gathering and gene name annotation

We performed data gathering and *gn* annotation as described in Vlachos and Gasperin, 2006. 16,609 biological article abstracts from papers curated in FlyBase were gathered, accompanied by the list of genes that were curated in their respective papers. The text was processed using the RASP toolkit tokenizer (Briscoe *et al.*, 2006) and for each abstract, the genes and their synonyms were tagged automatically using longest-extent pattern matching.

It is important to note that any errors made in this stage will propagate through all the steps of the process, affecting the quality of the training material created. The annotation of *gns* described above is bound to be imperfect. Gene names can be common English words resulting in false positives (Vlachos and Gasperin, 2006). Moreover, the lists of the genes curated for each paper do not contain all the *gns* appearing in its abstract. Nevertheless, given that training datasets created in this fashion were used successfully for *gn* recognition in Morgan *et al.*, 2004 and Vlachos and Gasperin, 2006 we expect that the noise will not affect the performance of the resulting system too detrimentally.

3.2 RASP and noun phrase extraction

For NP boundary detection we used a part-of-speech (POS) tagger and a syntactic parser to generate a list of grammatical relations (GRs) over the tokens. The POS tagger is a bigram HMM trained on general English text and has a module for handling unknown words. The parser is an unmodified parser for general English text. Both these modules are part of the RASP toolkit.

The tokens in the input are tagged in parallel by the POS-tagger and the dictionary-based *gn* tagger of the previous step. Then, any

*to whom correspondence should be addressed

[1] The corpus and detailed annotation guidelines can be found at http://www.cl.cam.ac.uk/ nk304/Project_Index

Table 1. Most frequent lemmas that are indicative of their class and their frequencies in the annotated material.

om ind.	protein	mutant	activity	expression	function	locus
f.	37	23	16	10	10	7

gm ind.	gene	element	transgene
f.	49	4	1

tokens that belong to *gns* are re-tagged as "singular proper name" in the POS tag stream. The resulting POS tag stream is parsed by RASP and the GRs generated are searched in order to recover NP boundaries. From the recovered NPs, we kept only those that contained one or more *gns*.

We evaluated the performance of the NP extraction module in order to assess the quality of the generated training material. Testing on the *mentions* of the test set, the module achieved 80.57% F-score (86.40% for left- and 89.22% for right-hand boundaries). During this experiment, we used the annotations of the *gns* produced by the dictionary-based tagger of the previous step, instead of the perfect manual annotation of the test set. While this produces more errors due to the imperfect *gn* recognition, it is a more realistic assessment of our training data creation technique. It is worth mentioning that about 25% of the errors arise from the parser failing to produce a complete parse of the sentence, in which case it produces partial parses that are more likely to contain errors.

3.3 Rule-based annotation

We have now created a corpus in which *gns* and *mentions* have been annotated automatically. In order to classify them in *gms* and *oms*, we developed a simple rule-based classifier using observations made on the manually annotated corpus and the annotation guidelines of Vlachos and Gasperin, 2006.

Two types of rules were created. The first type uses the tokens of the *mention* outside the *gn*. There are several tokens or pairs of tokens that are indicative of *gene-* or *oms*, such as *"gene"* or *"mutant"*. The tokens of the *gns* were not used because they are not expected to be indicative of mention's class.

We collected statistics on the frequency of the lemmas of tokens appearing in *mentions*. We kept noun or adjective lemmas only, according to their POS tag. Then we chose manually those that are indicative of each class, ignoring the labels of the test set. We kept the most frequent lemmas for each class and manually filtered out lemmas that appear accidentally in the *mention* without indicating its class. For example, *"fluorescent"* is removed because it is not indicative of the class, whereas *"protein"* is kept. The resulting lists appear in Table 1. The rule-based system classifies a *mention* in a certain class if it contains a token considered to be indicative of it.

The second type of rules uses the GRs output of RASP (Briscoe, 2006) and is applied after the first type. The key idea is that some *mentions* can be classified following an already classified mention in the appropriate syntactic context, expressed by the GRs. The form of a GR between two tokens with lemmas lem_1, lem_2, positions in the sentence idx_1, idx_2 and POS tags POS_1, POS_2 is:

$$type((lem_1, idx_1, POS_1), (lem_2, idx_2, POS_2))$$

Each rule checks for the existence of some RASP GRs and for mentions that participate in them in a specific role. For this purpose, we developed rules that take advantage of apposition, in which two NPs are placed next to each other and the second one (which is called appositive) defines or modifies the first. Two *mentions* in apposition are likely to belong to the same class.

In terms of RASP's GRs, apposition is expressed through the *text adjunct* relation (*ta*). The first rule developed takes advantage of the apposition between a pair of *mentions* in which one of the two has been classified by the rules that use indicative tokens. For example, consider the sentence:

... the localised activity of <om> the receptor kinase <gn> Torso </gn> </om> (<om> <gn> Tor </gn> </om>) at ...

"the receptor kinase Torso" is already classified as *om* because it contains the token "receptor". Given the GR:

$$ta((''kinase'', 19, NN1), (''Tor'', 22, NP1))$$

mention "Tor" should be classified as *om* as well.

The second rule takes advantage of the apposition between a *mention* and a token, whose lemma belongs to the lists of indicative tokens of Table 1. For example, the RASP analysis for the sentence:

... the protein, <om> <gn> RPS15 </gn> </om>, causes ...

includes the relation:

$$ta((''protein'', 13, NN1), (''RPS15'', 15, NP1))$$

"protein" is a lemma of the regular expression list for *oms* (see table 1). Hence, the *mention* "RPS15" should be classified as *om*, since it is in apposition with "protein". Finally, we created a version of the first rule to deal with cases where the appositive is coordination between two or more NPs.

Mentions not captured by any of the rules were not processed further nor were they added to the training material compiled. While we could attempt to increase coverage, we opted for accuracy and simplicity, since we did not want to build a classifier manually, but to bootstrap a statistical system from high precision data.

4 BOOTSTRAPPED SYSTEM

The bootstrapped system creates training examples using the steps described in section 3. These are used to extract features and train an SVM classifier. The latter is used to classify the *mentions* during testing combined with a post-processing scheme which is applied in order to take advantage of the coordination between *mentions*.

Support Vector Machines (SVM) (Vapnik, 1998) are state-of-the-art statistical learning models that perform classification and real valued function approximation tasks. They project the training data-points to a high dimensional feature space in which they find the separating hyperplane between two classes. During testing, SVMs project the data-points into the same higher dimensional feature space and classify them depending on the side of the separating hyperplane that they are found on. The SVM package used in this paper is SVM^{light} (Joachims, 1999). The features used to represent the *mentions* were extracted from two sources, either from the tokens or from the GRs output of RASP.

The tokens from inside the *mentions* but outside the *gn* are a generalisation of the indicative token rules of the rule-based classifier (Section 3.3). Since SVMs require numerical features as input, each token found in the *mentions* was lemmatized by RASP and was turned into a binary feature. Similar transformation was performed for the features that are described later in this section. In order to assess how useful tokens are as features, we checked the type of *mentions*,

in which each token of the test set appears. We found that 81.87% of the tokens appear in *oms* only, 11.70% in *gms* only and 6.43% in both classes. 93.57% of the tokens appear in one class only, so we expect tokens to be useful as features.

For the syntactic features, we used the GRs output of RASP. We expect GRs to be useful in cases where the context is vital to classify the *mentions* correctly, because GRs use information from the entire sentence. For each *mention* we added the following features:

- The tokens with which the *mention* is linked through apposition (Section 3.3).
- The tokens with which the *mention* is related via the preposition "of" as an argument. This helps to capture cases such as "the expression of torpedo", where the token "expression" indicates that "torpedo" refers to a gene.
- The verbs whose subject is the *mention* in a passive construction. This feature helps to capture occurrences such as "faf is expressed...", where the verb "express" indicates that "faf" refers to a gene.
- The verbs whose subject is the *mention*. The intuition behind this is that genes tend to be inactive, so *mentions* that are subjects of verbs such as "localise" are usually *oms*.
- The combination of the verb and the direct object for which the *mention* is the subject of the verb. This feature extends the previous one by further refining the action of the verb using its direct object, which can be useful to capture patterns like "hth encodes a homeodomain-containing protein", where the fact that "hth" encodes a protein suggests that "hth" refers to a gene. A special case of this rule was made for the verb "be", where the direct object GR is replaced by complement.

For each token related to a *mention* via one of the features described above a separate binary SVM feature is created using its lemma. For example, if the token "genes" is found to be in apposition with a *mention*, then a binary feature "gene_appositive" is created. If a lemma is related to *mentions* via more than one features, then a separate binary SVM feature is created for each case.

The instances of the training and the testing sets are transformed into vectors of their features. The SVM classifies each *mention* independently during testing. Consequently, mentions that are coordinated may not be classified in the same class. To avoid this, we post-process the output and identify the coordinated mentions using RASP's GRs. If two or more such *mentions* disagree, then the label of the *mention* that was classified with the highest confidence (larger margin assigned by the classifier) is assigned to all.

5 EXPERIMENTS

Table 2 presents statistics of the training corpus and of the test set. Since we consider both classes of equal importance and the task is not very imbalanced (32% *gms* - 68% *oms* in the test set), we decided to use accuracy, defined as the percentage of correctly annotated mentions over all mentions, as evaluation metric.

As expected, *mentions* containing lemmas that have been encountered in the training data are more likely to be correctly classified than those containing unknown lemmas. To obtain a more informative evaluation of the performance of the trained classifier, we present statistics for the seen, partially seen and unseen test *mentions* separately. For each *mention* we create a template consisting of the annotation tags and the lemmas of the tokens outside *gns*. Seen *mentions* are those whose template has been encountered in the training data. Partially seen *mentions* are those that although they are not seen, one or more of their lemmas have been encountered in the training data. They are counted towards *partially seen mentions in the same* or *in the opposite class* depending on whether their lemmas were in a training example of the same or the opposite class, respectively. Unseen *mentions* contain lemmas that have not been encountered in the training data. We expect the last two categories to be harder than the first two.

Another category of interest contains those *mentions* consisting only of one or more *gns*, which represent 49.39% of the test corpus (table 2). In these cases, only features based on the GRs can be extracted. We expect them to be the most difficult for our system to classify and we consider these separately in our evaluation.

Table 2. Corpora annotation statistics

	test corpus	training corpus
abstracts / sentences	81 / 600	16,609 / 111,822
tokens / avg. tokens per mention	15,704 / 2.54	2,923,188 / 2.58
mentions / *gn* only mentions	571 / 282	105,977 / N/A

Table 3. Evaluation results of the bootstrapped system.

	Accuracy		
Total / Unseen / Seen (same class)	78.63%	90.00%	98.88%
Partially seen (ind. / same / opp.) class	89.00%	89.83%	87.80%
Gene name only	61.70%		

5.1 Rule-based system results

Most *mentions* of the test set are *oms*. If all *mentions* are classified as *oms*, the accuracy would be 68.12%, giving a rather crude baseline. The rule-based system applied to the test corpus achieves 97.67% accuracy; the rules using the lemmas and the GRs achieve 97.98% and 94.7%, respectively. However, the coverage is low, at 38%. We didn't attempt to develop more rules, because our priority was for them to be simple and as accurate as possible, because they are used to create training examples for the bootstrapped classifier. To compare with the above baseline and the accuracy of the bootstrapped system, we considered all unclassified mentions of the test set as *oms*. This achieves an accuracy of 77.76% over all test *mentions*, substantially higher than the baseline.

5.2 Bootstrapped system results

The rule-based tagger tagged 32.17% of the *mentions* available in the training data; the rules using the lemmas and the GRs tagged 28.97% and 3.21%, respectively. This is lower than the 38% achieved on the test corpus, which can be attributed to the errors introduced by the automatic NP detection module used. Those left untagged were not used in the training of the SVM classifier. It is worth mentioning that many more *oms* than *gms* were generated (more than 3 to 1), resulting in a rather imbalanced dataset which could result in a biased classifier.

Table 4. Assessing the contribution of syntactic parser.

step(s) removed	total	unseen	seen	partially seen	gene name only
t	76.36%	70.00%	100.00%	88.60%	58.68%
f	76.71%	90.00%	98.88%	89.00%	57.80%
p	78.46%	90.00%	98.88%	89.00%	61.35%
t, f	72.68%	90.00%	100.00%	72.81%	56.94%
t, f, p	72.50%	90.00%	100.00%	72.81%	56.60%

Table 3 presents the evaluation of the bootstrapped system. For the seen *mentions*, performance is very high and comparable to the accuracy of the rule-based system in the instances that some rule is applied (97.67%). The performance in the partially seen and unseen categories was good as well, showing that the system is able to cope with *mentions* containing lemmas not encountered in the training data. It must be noted that the rule-based system has no way of dealing with cases that do not contain lemmas covered by its rules.

For mentions that include only *gns*, the accuracy was lower (61.70%). While this was expected, since such mentions do not contain any tokens to use their lemmas as features, we hoped that the syntactic features would be able to deal with them. However, in many cases no syntactic features were extracted because the rules could not take advantage of the GRs generated by the parser. This can be partially attributed to the fact that the parser used is a general English parser, therefore its performance is likely to be worse than expected in texts from the biomedical domain. While the accuracy in these instances is low, it is still better than the accuracy achieved by considering all the instances as *oms* (56.74%).

We were interested in assessing the contribution of the syntactic parser to the performance of the booststrapped system. The parser is used at three points throughout the system. During data generation, we take advantage of the apposition to generate more training examples (Section 3.3). Then, we extract features based on GRs (Section 4). Finally in the post-processing stage, in order to correct coordinated *mentions* that are annotated inconsistently, we removed the steps to assess their importance (Table 4).

The generation of training examples (row "t") and the syntactic features (row "f") contribute more significantly than the post-processing (row "p"). The removal of post-processing reduces the accuracy from 78.63% to 78.46%, while the removal of each one of the other steps decreases the performance by approximately 2%, which is mainly due to decreased ability to classify *gn* only *mentions*. Due to the same reason, removing both these steps reduces the performance further to 72.68%. These results are rather predictable, since the syntactic features and the examples generated by the apposition rules enable the bootstrapped system to classify this class of *mentions* (49.39% of our test set). As expected, the classifier is biased towards the *om* class. We experimented with the j-trick (Morik *et al.*, 1999) and the use of RBF kernels, but the results did not improve.

6 RELATED WORK AND CONCLUSION

We presented an approach to determine whether an NP containing one or more *gns* refers to an actual gene or to another entity. We developed a bootstrapped system based on syntactic parsing and SVMs. The training data were automatically annotated by a rule-based tagger, whose rules were semi-automatically created.

There is some related work in the literature of biomedical text mining. For example, Hatzivassiloglou *et al.*, 2001 built a machine learner to recognise names of genes, proteins and RNA labels. They noted though their performance suffered due to annotation inconsistencies. Similar observations were made by Dingare *et al.*, 2005 concerning the annotated data that were used for the BioNLP2004 shared task (Kim *et al.*, 2004). As mentioned earlier, a lot of these problems are due to the way *gns* are used to refer to entities that are not necessarily genes.

In our approach, we formulate the task in two stages. First we identified the *gns* and then we disambiguated their class. This is similar to the way the Entity Detection and Tracking task is formulated in the ACE projet (ACE, 2004). Apart from improving inter-annotator agreement for the task, it also results in more informative output. The main advantage of our system is that it does not require manually annotated training data. It uses a small set of simple, manually created rules to generate training examples out of a big, unannotated corpus. The bootstrapped system achieves higher performance than the rule-based one. As a disadvantage, we used the same set for rule development, feature coverage estimation and testing. Ideally, we should have used a different one for testing but (to our knowledge) there are no other manually annotated corpora.

The syntactic parser contributes substantially to the achieved performance. It was also observed that many errors occur in examples where no syntactic features are extracted. Therefore, the performance could be improved by adding more syntactic features. An extension to this work is to attempt identifying more types of entities referred to by *gns*, such as proteins and to use the system to deal with similar tasks in other domains. Furthermore, this system could serve as a module in a more complicated task, such as coreference resolution and it would be of interest to assess its potential in such context.

REFERENCES

ACE (2004). Annotation guidelines for entity detection and tracking.

Blaschke, C., Hirschman, L., and Yeh, A., editors (2004). *Proceedings of the BioCreative Workshop*, Granada.

Briscoe, T. (2006). An introduction to tag sequence grammars and the rasp system parser. Technical Report UCAM-CL-TR-662, University of Cambridge.

Briscoe, T., Carroll, J., and Watson, R. (2006). The second release of the rasp system. In *Proceedings of the COLING/ACL 2006*, Sydney, Australia.

Dingare, S., Nissim, M., Finkel, J., Manning, C., and Grover, C. (2005). A system for identifying named entities in biomedical text: how results from two evaluations reflect on both the system and the evaluations. *Comparative and Functional Genomics*, **6**(1-2), 77–85.

Hatzivassiloglou, V., Duboué, P. A., and Rzhetsky, A. (2001). Disambiguating proteins, genes, and rna in text: a machine learning approach. In *ISMB*, pages 97–106.

Joachims, T. (1999). Making large-scale support vector machine learning practical. *Advances in kernel methods: support vector learning*, pages 169–184.

Kim, J.-D., Ohta, T., Tsuruoka, Y., Tateisi, Y., and Collier, N., editors (2004). *Introduction to the bio-entity recognition task at JNLPBA*, Geneva, Switzerland.

Morgan, A. A., Hirschman, L., Colosimo, M., Yeh, A. S., and Colombe, J. B. (2004). Gene name identification and normalization using a model organism database. *J. of Biomedical Informatics*, **37**(6), 396–410.

Morik, K., Brockhausen, P., and Joachims, T. (1999). Combining statistical learning with a knowledge-based approach - a case study in intensive care monitoring. In *Proceedings of ICML-99*, pages 268–277. Morgan Kaufmann, San Francisco, CA.

Vapnik, V. N. (1998). *Statistical Learning Theory*. Wiley-Interscience.

Vlachos, A. and Gasperin, C. (2006). Bootstrapping and evaluating named entity recognition in the biomedical domain. In *Proceedings of the HLT-NAACL BioNLP Workshop on Linking Natural Language and Biology*, pages 138–145, New York.