Exploring Match Scores to Boost Precision of Gene Normalization

Cheng-Ju Kuo[1] cju.kuo@gmail.com
Yu-Ming Chang[2] porter@iis.sinica.edu.tw
Han-Shen Huang[2] hanshen@iis.sinica.edu.tw
Kuan-Ting Lin[2] woody@iis.sinica.edu.tw
Bo-Hou Yang[2,3] ericyang@iis.sinica.edu.tw
Yu-Shi Lin[2] bathroom@iis.sinica.edu.tw
Chun-Nan Hsu[2] chunnan@iis.sinica.edu.tw
I-Fang Chung[1] ifchung@ym.edu.tw

[1] Institute of Bioinformatics, National Yang-Ming University, Taipei, Taiwan
[2] Institute of Information Science, Academia Sinica, Taipei, Taiwan
[3] Department of Electrical Engineering, Chang-Gang University, Tao Yuan, Taiwan

1 Introduction

Gene normalization task is to identify EntrezGene IDs corresponding to the human genes and direct gene products appearing in a given MEDLINE abstract. Given a dictionary that maps gene and protein synonyms to EntrezGene IDs, a naive approach to the problem is to apply a gene mention tagger to identify all potential name entities of genes and then look them up in the dictionary. However, mostly due to the difficulty to compile a complete yet noise-free dictionary for gene synonyms [5], the results are far from satisfactory. In our experiments using a gene mention tagger based on a conditional random field (CRF) [4] model and a string matcher based on softTFIDF [1] to look up the dictionary, the F-score is below 0.5. To improve the performance, previous work proposed many methods to clean up dictionaries. These methods may help case by case but may not applicable in general. In this paper, we focus on the problem of whether there exists a systematic method that always improves the result of dictionary lookup. We propose to train an ensemble of classifiers using AdaBoost [2] to recognize true positives from false ones based on the match scores, which are readily available when anyone applies an approximate string matching function to look up the dictionary. Experimental results show that applying boosting can successfully increase the F-score from about 0.56 to 0.69 with our best F-score reaching 0.75. These results were obtained without modifying the dictionary.

2 Method and Results

Given an abstract, our system takes the following three steps to return the EntrezGene IDs mentioned in the abstract:

1. We directly apply our gene mention tagger from the GM Task [3] to identify possible entities of gene names.
2. For each entity, we apply an approximate string matching function to compare the entity with all entries in the dictionary. Each entry contains the EntrezGene ID and the synonyms of a human gene. A list of top ten match scores is returned with the ID of the top match.
3. Based on the match scores, an ensemble of classifiers is applied to determine if the top ID actually corresponds to the entity. If positive, the ID with its score will be returned; otherwise, the result will be discarded.

We describe the details of these steps as follows. Our gene mention tagger is a union of bidirectional parsing CRF models from the GM Task [3]. This tagger was trained by the 15,000 training examples from the GM Task, which contains data of gene/protein names of all species. But we only need human gene/protein names in the GN Task. As a result, though this tagger achieves a 0.8658 F-score for the GM Task, its F-score for the training data of the GN Task is far from this level. However, since it is not necessary to identify all gene mentions in abstracts, before a training data set for human genes is available, we will have to settle with this gene tagger.

Next, we applied TFIDF and softTFIDF [1] to compute the similarity between a tagged gene name entity and a synonym in the dictionary. A preprocessing step that transforms a string into a token vector was applied in advance when we applied TFIDF, including case normalization, replacement of hyphens with blanks, removal of punctuation symbols and parenthesized strings, etc. The idea is to increase the chance of matchings. For softTFIDF, there is no need to perform the preprocessing step because we have Jaro and Jaro-Winkler with TFIDF to tolerate slight difference between terms in gene names. We assigned a threshold δ to filter the outputs of the dictionary lookup. If the highest match score is less than δ, the tagged entity in the abstract will be discarded. Otherwise, the ID with the top score will be returned as an answer. This is how we obtained the results of Step 2 in Table 1.

The feature vector for our ensemble classifier is derived from the top ten match scores of synonyms of ten distinct genes. Let $(s_1, s_2, \ldots, s_{10})$ be the top ten scores, the feature vector consists of twelve features defined as follows:

$$(s_1, s_1 - s_2, s_2 - s_3, \ldots, s_9 - s_{10}, s_1 - s_{10}, \mathrm{Var}(s_i)).$$

The idea is to characterize the distribution of the match scores to discriminate a false positive. This feature set assumes that the dictionary contains entries that share many terms such that an entity in an abstract may match many synonyms in the dictionary. We applied AdaBoost to train an ensemble classifier with this feature set because boosting can automatically take advantage of the fact that these features are not equally important. We stopped iterations of AdaBoost at thirty because ensembles with thirty decision stumps performed the best in our experiments. Suppose the accuracy of our classifier is α, then the new F-score after applying our classifier will be:

$$P = \frac{TP \cdot \alpha}{TP \cdot \alpha + FP(1-\alpha)}, \quad R = \frac{TP \cdot \alpha}{TP + FN}, \quad F = \frac{2PR}{P+R}.$$

Therefore, if we have a dictionary lookup result whose FN is small but TP and FP are large (i.e., low precision and high recall), then our classification method will boost the precision as well as the F-score.

When more than one entry in the dictionary share a top match score, our feature set would be insufficient to recognize which entry is a true positive. In this case, we have two tie-breaking (TB) strategies to handle the situation. One is to simply discard that entity to reduce false positives. The other is to return the ID of the entry that maximizes the occurrences that the entity appears as a substring in the synonyms of that entry. The rest will be sent to the classifier for further filtering.

Table 1 shows the results of our experiments. All trials used the output of our CRF tagger as the input. Due to the time constraint, we only had the result of the configuration – TFIDF and $\delta = 0.5$ before the deadline. Apparently, the threshold is too low so that Step 2 passed many false positives to Step 3. However, our classifier still successfully filtered most of them to improve the F-score from 0.56 to 0.69. After the deadline, we raised the threshold to decrease false positives by Step 2 and the F-scores went up. In our experiments with softTFIDF, since approximate string matching was used to compute the similarity, the scores are usually higher than TFIDF. Therefore, higher thresholds are necessary for softTFIDF. We found that the F-score was increased as we increased δ but when $\delta = 0.99$, Step 3 failed to boost the F-score because in these cases, it is recall that needs boosting rather than precision. Nevertheless, our best F-score was achieved when Step 3 was applied to boost Step 2 with $\delta = 0.95$ and our tie-breaker applied.

Table 1: Performance Comparison for Gene Normalization.

Method	δ	Step2 (Precision/Recall/F-score)	Step3
TFIDF	0.5	0.4523/0.7375/0.5607	0.7166/0.6636/0.6891(* submitted)
	0.9	0.6495/0.6777/0.6633	0.8402/0.6229/0.7154
	0.95	0.6904/0.6714/0.6714	0.8637/0.5974/0.7063
softTFIDF (Jaro)	0.9	0.6163/0.7286/0.6678	0.8155/0.6700/0.7356
	0.95	0.7496/0.6866/0.7167	0.8484/0.6560/0.7399
	0.99	0.8210/0.6547/0.7285	0.8539/0.6331/0.7271
softTFIDF (Jaro-Winkler)	0.9	0.4670/0.7503/0.5757	0.7524/0.6968/0.7235
	0.95	0.6389/0.7235/0.6786	0.7957/0.6751/0.7305
	0.99	0.8077/0.6636/0.7286	0.8341/0.6407/0.7247
softTFIDF+TB (Jaro)	0.9	0.5907/0.7630/0.6659	0.7890/0.7006/0.7422
	0.95	0.7159/0.7222/0.7209	**0.8256/0.6878/0.7505**
	0.99	0.7918/0.6929/0.7391	0.8328/0.6662/0.7402
softTFIDF+TB (Jaro-Winkler)	0.9	0.4555/0.7834/0.5761	0.7315/0.7324/0.7320
	0.95	0.6172/0.7579/0.6803	0.7699/0.7121/0.7399
	0.99	0.7779/0.7006/0.7372	0.8027/0.6789/0.7356

References

[1] Cohen, W. W., Ravikumar, P., and Fienberg, S. E. A comparison of string distance metrics for name-matching tasks. In *Proceedings of the IJCAI-2003 Workshop on Information Integration of the Web*, 2003.

[2] Freund, Y. and Schapire, R. E. A decision-theoretic generalization of on-line learning and an application to boosting. In *European Conference on Computational Learning Theory*, pages 23–37, 1995.

[3] Kuo, C.-J., Chang, Y.-M., Huang, H.-S., Lin, K.-T., Yang, B.-H., Lin, Y.-S., Hsu, C.-N., and Chung, I.-F. Rich feature set, unification of bidirectional parsing and dictionary filtering for high f-score gene mention tagging. Sumbitted to Second BioCreAtIvE Challenge Workshop, 2007.

[4] Lafferty, J., McCallum, A., and Pereira, F. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. 18th International Conf. on Machine Learning*, pages 282–289. Morgan Kaufmann, San Francisco, CA, 2001.

[5] Morgan, A. A., Wellner, B., Colombe, J. B., Arens, R., Colosimo, M. E., and Hirschman, L. Evaluating the automatic mapping of human gene and protein mentions to unique identifiers. In *Pacific Symposium on Biocomputing*, pages 281–291, 2007.