Report on Gene mention extraction

Ruoyao Ding
Computer and Information Science Department
University of Delaware
Newark Delaware 19716
ryding@udel.edu

Abstract

Gene mention (GM) extraction is an important initial step for many bio-mining tasks. Due to the problems of name variation and name ambiguity, the GM extraction task is particularly challenging. We developed a pivot based GM tagger to solve those two problems, we believe our tagger will have a better performance in respect of providing gene normalization input, comparing with the current existing GM taggers.

1. Introduction

In recent years, as the amount of biological literature increased rapidly, how to extract the desired information form the literature has been given more and more effort. Extracting gene mentions is an important initial step for many bio-mining tasks.

The fact that authors rarely use standardize gene names (name variation problem) and gene short names naturally share the same morphology with other types of entities (name ambiguity problem) makes gene mention extraction task particularly difficult.

A large body of work has been done for GM extraction, and many GM taggers are already available to use (Settles,B, 2005), (Heinz JF et al., 2007), (Hsu CN et al., 2008). Most of them fail to extract gene names with complex morphology, and some only perform well in some specific domains.

We developed a pivot based GM tagger, which uses a pivot based dictionary matching method as an initial tagging step, and a pivot based disambiguation method to get the final gene mention output.

In section 2, we will introduce the methods used in our GM tagger. The pros and cons of our methods will be discussed in section 3, some initial test result will also be shown in this section. Section 4 will be the conclusion.

2. Methods

One gene name can be divided into three parts: prefix, pivot, and suffix. The prefix is usually some description of the actual gene name, e.g., the species information which the gene name belongs to. The suffix is usually number and Greek number, which can be used to identify one specific gene name in one gene family. The rest part of the gene name is the pivot, which stores the gene name sense. One example of dividing one gene name into prefix, pivot, and suffix is shown in Figure 1, and the pivot stemming algorithm to get pivot and suffix from one name is shown in Figure 2.

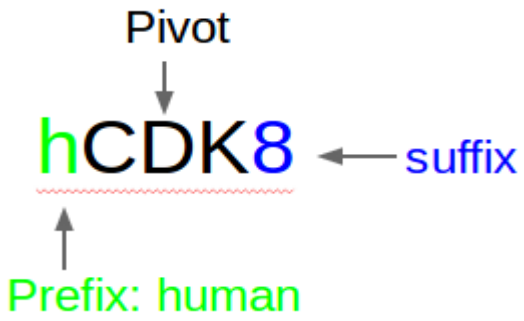

Figure 1: Example of pivot dividing

Define three types of key elements in one name: letter, number, Greek munber.

Input one name string S, separate each pair of adjacent different types of elements with space, replace hyphen with space. Elements from left to right are: $E_0, E_1, \ldots, E_n$.

```
for i from 0 to n:
  if E_i is number or Greek number
    move E_i to the end of the name
  else
    break

for i from n to 0:
  if E_i is number or Greek number
    next
  else
    break

if i eq n
  get element from 0 to n as pivot, set suffix =
  NO
else
  get element from 0 to i as pivot, element i+1
  to n as suffix
```

Figure 2: pivot stemming algorithm

The workflow of our tagger is shown in Figure 3. We first generate simply name candidates from the input text, then tag the gene candidate in those name candidates using a pivot based dictionary matching method, two disambiguate models are used to get the gene pivot, and finally get the gene names based on the gene pivots.

2.1 Dictionary

Our initial dictionary is built from UniProt database reviewed part. We extract all names (recommend full names & short names, alternate full names & short names, synonyms) in species: human, mouse, rat, and Arabidopsis thalina. We believe the entries in these four species could cover most of the gene names, the dictionary will be enhanced to include more species in the future if needed.

We build a gene pivot dictionary based on the initial dictionary: for each original name in the initial dictionary, use pivot stemming algorithm to get the pivot and suffix, the new entry in the pivot dictionary uses pivot as key, suffix, original name, and full name as value.

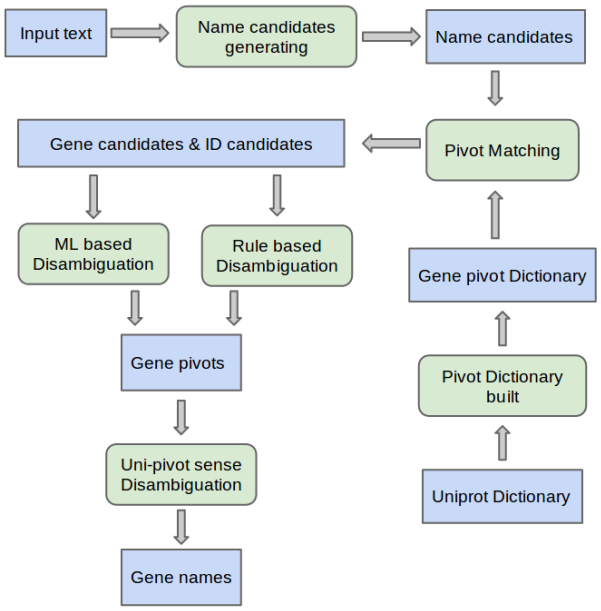

Figure 3: Workflow of our GM extraction system

2.2 Name candidates generating

Some of the names shown in the text have complex morphologies, e.g., interleukin (IL)-4, we cannot match this name with any dictionary entry directly.

A set of rules are applied to address this problem: from one name with complex morphology, get proper name candidates which can be used to match with the gene dictionary (Table 1).

Parentheses rule
interleukin (IL)-4 → interleukin-4 & IL-4
Suffix in conjunction rule
Protein kinase C alpha, beta, and gamma → Protein kinase C alpha & Protein kinase C beta & Protein kinase C gamma
Suffix in set rule
ERK 1-8 → ERK 1 & ERK 2 & …& ERK 8

Table 1: Name candidates generating rules

2.3 Pivot matching

In this step, we match the name candidates with dictionary entries to get gene candidates. Here we propose a pivot based dictionary matching algorithm (Figure 4), to solve the name variation problem.

input one name candidate t, get its pivot P_t and suffix S_t using pivot stemming algorithm

```
if exists key(P_t) in the dictionary, get value
   of key(P_t), S_d
   if S_t eq S_d
     return full match gene candidate t
   else if same_type(S_t, S_d)
     return pivot match gene candidate t
   else if S_t eq NO & S_d ! eq NO
     return family match gene candidate t
   else
     retun not match
else
   not match
```

Figure 4: Pivot matching algorithm

2.4 Pivot disambiguation

Due to the fact that gene short names naturally share the same morphology with other types of entities, when one name in the text matched with one short name entry in the gene dictionary, we cannot say that name is a gene, future name sense disambiguation is needed.

One type of name, called functional term (F-terms), is quite helpful for identifying the gene name (.M. Narayanaswamy et al., 2003). Names ending with F-term are believed to have gene sense.

Context information is also quite helpful for disambiguating names. As some of the names may not have context information which can be used for disambiguation, instead of directly disambiguating the names, here we try to disambiguate the name pivot, and all the names that share the same pivot in the same article will get the same name sense.

Two pivot disambiguation modules are used to get the gene pivot.

2.4.1 ML based disambiguation

As the current existing GM corpuses are usually either domain specific, or not large enough to be used to train one disambiguation model which can be used in general. Here we proposed a semi-auto training set build method, trying to get one training corpus which is large enough to be able to produce a general gene disambiguation model.

We first get all gene short names from Entrez Gene dictionary, and get all the PMIDs which contain those short name, as well as their full names, then based on the full names we get positive instances and negative instances: if the full name is ending with an F-term or fully matched with our gene dictionary, the name is viewed as a positive instance; if the full name doesn't get a uni-gram bag of word match with any entry in our gene dictionary, the name is viewed as a negative instance. Last, all the positive full names and short names are replaced with a word "NAMEP", all the negative full names and short names are replaced with a word "NAMEN", so beforehand information is removed. In this way, we get 30,000 positive instances and 30,000 negative instances in 60,000 PMIDs for training, 20,000 negative instances and 20,000 negative instances in 40,000 PMIDs for model

testing.

A SVM model is trained using the context word feature, we set a margin of two words, choose uni-gram and bio-gram context words of each instance as features.

2.4.2 Rule based disambiguation

Some context information is quite helpful for name disambiguation, but usually difficult to be annotated, thus difficult to capture by a ML model. Three rules are added to capture those kinds of context information.

If one gene candidate has a full name in the same article, this would be a perfect clue which can be used in disambiguation. If the full name is matched with the dictionary, or has an F-term ending, both the full name pivot and short name pivot are assigned with gene sense.

Appositive is another good clue which can be used in disambiguation. If one gene candidate has an appositive and that appositive is ending with an F-term, then the gene candidate pivot is assigned with gene sense.

Gene names shown in the same text should not be isolated with each other, we assume that those names should have some kind of relation. Based on this assumption, we make the gene candidates in relation rule: if some gene candidates are shown in conjunction, all their pivots are assigned with gene sense; if two gene candidates are connected with word "or" or slash, both of their pivots are assigned with gene sense; if one gene candidate and its synonyms appear in the same article, all their pivots are assigned with gene sense.

2.5 Pivot based name disambiguation

Based on the pivot sense information, we assign all the names that share the same pivot in the same text the same name sense (whether gene or not) with the pivot, and output all the names with gene sense.

3. Discussion

We use a dictionary matching method as an initial step to get gene candidates, this should produce a higher precision, as we can get a clear cut name boundary and filter out some general names which share the same context with gene.

However, there are also two problems: first, authors rarely use the standardize names, too many name variations makes it's particularly difficult to match the name in the text with the name in the dictionary. Second, there is no dictionary which can cover all the gene names, as there are always new genes come out, and some existing genes may change their names. Our pivot matching algorithm is trying to solve the problem of name variations, as well as the problem of dictionary incomplete.

If the final goal is gene normalization, the dictionary matching part is essential, that's probably the reason why some pure ML-based GM tagger can have a good performance in the GM evaluation, but when they are used as gene normalization input, the performance is not well.

In the disambiguation part, we just use context feature within distances of two words, the reason we didn't use a larger distance is that we noticed usually the left two words and right two words would be enough for disambiguate one name. If we enlarge the margin, the recall will be improved, well precision will drop. Instead, we used a pivot based disambiguation method to improve the recall, this would get even more benefit when the input text is full length, as more names share the same pivot will be there.

We also use three rules to help future improve

the disambiguation performance: the acronym, the relative clause, and gene candidates in relation, these features are not difficult to capture, but it's difficult to get a training corpus which have those information annotated, so ML model can hardly capture these features.

We performed an initial test on 100 PMIDs from BioCreative 2 GN corpus, those PMIDs are safe to use because we didn't train on this corpus. 300 gold standards are given in these 100 PMIDs, the result of our tagger is: precision = 216/293 = 0.74, recall = 216/300 = 0.72, final F-score is 0.73. After analyzing the result we found that some FPs are actually gene names, they just weren't tagged by BioCreative annotators, and some of gold standards are just the descriptions of the actual gene names, this result in many FNs. So the result shown here is not precise, future evaluation is needed to show the performance of our system.

4. Conclusion

We propose a pivot based GM tagger. Instead of processing the names, our tagger is focus on the pivots. Names with complex morphology are changed into simply names. The pivots matching algorithm is proposed to solve the problem of name variation and dictionary incomplete. Pivot based disambiguation method is used to cover the names which do not have enough context information for disambiguation.

References

Kim JD, Ohta T, Tateisi Y, Tsujii J. 2003. GENIA corpus-a semantically annotated corpus for bio-textmining. Bioinformatics.

Morgan AA, Lu Z, Wang X, Cohen AM, Fluck J, Ruch P, Divoli A, Fundel K, Leaman R, Hakenberg J, et al. 2008. Overview of BioCreative II gene normalization. Genome Biology.

Wilbur,J. et al. 2007. Biocreative 2. gene mention task. In Proceedings of the Second BioCreative Challenge Evaluation Workshop, Centro Nacional de Investigaciones Oncologicas (CNIO), Madrid, Spain. pp. 7–16.

Settles,B. 2005. ABNER: an open source tool for automatically tagging genes, proteins, and other entity names in text. Bioinformatics, 21, 3191–3192.

Hsu CN, Chang YM, Kuo CJ, Lin YS, Huang HS, Chung IF. 2008. Integrating High Dimensional Bi-directional Parsing Models for Gene Mention Tagging. Bioinformatics.

Heinz JF, Mevissen T, Dach H, Oster M, Hofmann-Apitius M. 2007. ProMiner: Recognition of Human Gene and Protein Names using regularly updated Dictionaries. The Second BioCreative Challenge Evaluation Workshop.

.M. Narayanaswamy, K. E. Ravikumar, K. Vijay-Shanker. 2003. A Biological Named Entity Recognizer, Pacific Symposium on Biocomputing 8: 427-438.

Peng Y, Tudor C, Torii M , Wu CH, Vijay-Shanker K. 2012. iSimp: A Sentence Simplification System for Biomedical Text. IEEE International Conference on Bioinformatics and Biomedicine (BIBM2012). pp. 211-216.