Original article

Unsupervised gene function extraction using semantic vectors

Ehsan Emadzadeh*, Azadeh Nikfarjam, Rachel E. Ginn and Graciela Gonzalez

Department of Biomedical Informatics, Arizona State University, AZ 85259, USA

*Corresponding author: Tel: 480-884-0220; Fax: 480-884-0239; Email: eemadzad@asu.edu


Received 31 January 2014; Revised 17 July 2014; Accepted 23 July 2014

Abstract

Finding gene functions discussed in the literature is an important task of information extraction (IE) from biomedical documents. Automated computational methodologies can significantly reduce the need for manual curation and improve quality of other related IE systems. We propose an open-IE method for the BioCreative IV GO shared task (subtask b), focused on finding gene function terms [Gene Ontology (GO) terms] for different genes in an article. The proposed open-IE approach is based on distributional semantic similarity over the GO terms. The method does not require annotated data for training, which makes it highly generalizable. We achieve an F-measure of 0.26 on the test-set in the official submission for BioCreative-GO shared task, the third highest F-measure among the seven participants in the shared task.

Database URL: https://code.google.com/p/rainbow-nlp/

Introduction

Mining biomedical literature aims to reduce manual labor and provide enriched information that can empower advances in medical research and treatments. Lu et al. (1) demonstrated that there is an increasing interest to use text-mining techniques for curation workflows. Currently, literature curation is challenged by a lack of automated annotation techniques, particularly for Gene Ontology (GO) annotations (1). In medical informatics alone, the number of indexed articles has increased by an average of 12% each year between 1987 and 2006 (2, 3), with close to 20 million articles indexed in PubMed in 2013. With an increasing number of publications detailing complex information, the need to have reliable and generalizable computational techniques increases rapidly.

Finding gene functions discussed in literature is crucial to genomic information extraction (IE). Currently, tagging the gene functions in published literature is mainly a manual process. Curators find gene function evidence by reviewing each sentence in relevant articles and mapping the results to standard ontologies, and, specifically for this problem, to the GO (4), a controlled vocabulary of gene functions. The BioCreative IV GO workshop (5) aims to automate gene function curation through computational methods. With a focus on gene functions, it includes two subtasks: (i) retrieving GO evidence sentences for relevant
genes and (ii) predicting GO terms for relevant genes. We focus on subtask b, which finds the related gene functions (GO terms) in a set of genes discussed in an article. More detail about the shared task and the corpus can be found in Auken et al. (6). This task is similar to BioCreative I subtask 2.2, which was held in 2004 (7). Blaschke et al. (7) summarized the results for BioCreative I. For subtask 2.2, the highest precision was reported to be 34.62% (8). BioCreative IV GO subtask 2 includes an annotated corpus to enable measurement of recall and F-measures. Couto et al. (9) used an information retrieval technique to find related sentences and GO terms. Chiang et al. (8) combined sentence classification with pattern mining. Ray et al. (10) proposed a solution based on probabilistic model and naive Bayes classifier. Most of the participants in the previous related task focused on information content and statistical models combined with machine learning. Here, we propose an unsupervised method based on distributional semantic similarity that can be easily applied for different types of texts and ontologies.

We decided to apply an unsupervised method to see how well semantic similarity methods without any machine-learning or engineered features could perform compared with existing methods. If the performance is on par with the supervised methods, then we can benefit from unsupervised technique to leverage the performance of the supervised methods. In addition, finding gene functions discussed in a document seems a good application for semantic similarity, enabling comparison of different semantic similarity methods. None of the previous work in BioCreative I used semantic similarity methods, including vector- or graph-based methods. Our proposed technique is completely unsupervised, based solely on semantic similarity without training on the provided data set; this characteristic makes the method unlikely to overfit the data set and generalizable to the extraction of any major concepts mentioned in a document. The proposed method achieved the third highest F-measure among the seven participants in the shared task.

**Material and methods**

Our method is based on distributional semantic similarity of sentences to GO terms. We use semantic vectors package (11) implementation of latent semantic analysis (LSA) (12) with random indexing (13) to calculate semantic similarities. GO terms’ semantic vectors are created based on GO names defined in GO; one semantic vector is created for each term in the ontology. Stop words are removed from GO name, and they are generalized by Porter stemming (14).

Figure 1 shows the overall flow of our proposed method. After creating GO semantic vectors, the objective is to find whether a sentence is related to a gene. We do this by using lexical patterns and generalizing the sentence and gene symbol (e.g. removing the numbers and...
nonalphabetic characters). If ‘Sentence Gene Matcher’ predicts that a sentence is related to a gene, then we calculate its semantic similarity to all GO terms using already generated semantic vectors. The articles are provided in BioC format (15) in which sentences, passages and the types of passages (heading, paragraph, etc.) are identified. The ‘Go Finder’ module finds all related GO terms to the sentence and generates the triplet of sentence, gene and GO term. Finally, the shared task expected output format is generated by ‘BioC output generator’. In the next section, we explain the ‘GO Finder’ module in more detail.

Semantic similarity

LSA with random indexing is used for calculating semantic similarity. LSA is a vector-based semantic similarity method that applies dimension reduction on document-term matrix before calculating cosine similarity of two terms.

The original proposed LSA algorithm uses singular value decomposition (SVD) for dimension reduction (12). This is a computationally expensive algorithm. Random indexing (13) technique has shown to be as effective as SVD but with linear complexity (16, 17). The semantic vectors are created for all GO terms regardless of their position in the GO graph. For creating document-word matrix, we consider each GO concept as a document and use the name field (‘GO term’) in GO for extracting terms of the node. A semantic vector is created for each concept in the ontology. For extracting terms from GO names, they are preprocessed by removing stop words (e.g. ‘the’, ‘a’), and then the extracted terms are generalized by Porter stemming (14).

‘GO Finder’ module

GO Finder finds related GO terms for each sentence. We define G as a set of top m GO terms with highest semantic similarity to the sentence. D is the set of top n GO terms with high similarity to the abstract of the related article. The following function returns top k similar GO terms for a given query:

\[
\text{TopSimilarGO}(query, k) = \{ x | x \in \text{GOTerms} \land \{ y \in \text{GOTerms} \mid \text{Sim}(x, query) < \text{Sim}(y, query) \} < k \}
\]

And G and D sets are

\[
G(\text{sentence}) = \text{TopSimilarGO}(\text{sentence}, m) \\
D(\text{abstract}) = \text{TopSimilarGO}(\text{abstract}, n)
\]

If a sentence is predicted to have the gene mention, the predicted GO terms for the sentence and gene are the conjunction of top similar GO terms to the sentence (set G) and top similar GO terms to the related abstract (set D):

\[
\text{GeneGO}(\text{gene}, \text{sentence}, \text{abstract}) = \{ G(\text{sentence}) \cap D(\text{abstract}) \} \text{ if } \text{HasGene}(\text{sentence}, \text{gene}) \text{ else } \{
\}
\]

A GO term with the highest semantic similarity to the sentence in the GeneGO set will be chosen as the final GO annotation for each gene in the sentence. For example, if a sentence top \(m = 2\) similar GO terms are \(\{g_5, g_{10}\}\) and the abstract top \(n = 5\) GO terms are \(\{g_4, g_8, g_5, g_2, g_9\}\), then the final predicted GO terms for the sentence related to the gene will be \(g_5\). The tuning parameters \(m\) and \(n\) control precision and recall.

Table 1 summarizes the number of sentences in the training set that were detected by ‘Sentence Gene Matcher’ as relevant to a gene and also annotated to have a gene function. The table shows that ‘abstract’, ‘front’ and ‘title2’ sections of each document are the most important sections that can include gene function. The passage types appearing in Table 1 are taken exactly from the corpus. Table 2 shows an example for each passage types from publications in the train set. We found that the first sentences of paragraphs have information about GO terms, but including all sentences in a paragraph will significantly reduce the precision. Therefore, we limit searching for the gene functions to the mentioned sections of the article. We choose one set of values for \(m\) and \(n\), for ‘Front’, ‘Abstract’ and ‘Title2’ (\(mFAT, nFAT\)), and choose a different set for the first sentence of the paragraphs (\(mParagraph, nParagraph\).
Evaluation method

The evaluation method is explained in detail by Mao et al. (5). The GO terms predicted by the system are compared with gold standard to calculate precision, recall and F-measure. In addition to exact match, hierarchical precision, recall and F-measure are used to evaluate the system. In the hierarchical evaluation method, all of the ancestors of an annotated GO term in gold standard and system output are used to calculate the precision and recall. Hierarchical measures are calculated using the following formula below, where Predicted\textsubscript{expanded} and Gold\textsubscript{expanded} are system output and gold standard annotations expanded with the ancestor of selected GO terms in the ontology.

\begin{align*}
P &= \frac{|\text{Predicted}_{\text{expanded}} \cap \text{Gold}_{\text{expanded}}|}{|\text{Gold}_{\text{expanded}}|}, \\
R &= \frac{|\text{Predicted}_{\text{expanded}} \cap \text{Gold}_{\text{expanded}}|}{|\text{Predicted}_{\text{expanded}}|}
\end{align*}

Results and discussion

Tuning parameters

To achieve the highest F-measure, the tuning parameters \((m \text{ and } n)\) need to be adjusted accordingly. We use two sets of values for \(m\) and \(n\); one set for the first sentence of each paragraph (\(m_{\text{Paragraph}}\) and \(n_{\text{Paragraph}}\)) and another for FAT passage types (\(m_{\text{FAT}}\) and \(n_{\text{FAT}}\)). To find the best tuning parameters, we evaluate the system with different values for a particular parameter while values of other parameters are constant. The experiment is repeated for all four parameters. Figure 3 shows variation of performance when tuning parameters change. Overall when parameters increase, precision increases and recall decreases. We tried to find the values that yield maximum F-measure. Figure 3a depicts precision, recall and F-measure change in respect to \(m_{\text{FAT}}\) changes. As \(m_{\text{FAT}}\) increases, precision declines and recall increases.

The maximum F-measure is achieved for \(m_{\text{FAT}} = 9\). Therefore, we assign \(m_{\text{FAT}}\) to 9, and try to find the best value for \(m_{\text{Paragraph}}\). Figure 3b shows the change of performance based on change of \(m_{\text{Paragraph}}\) and best result achieved for \(m_{\text{Paragraph}} = 15\). Figure 3c shows variation of performance when \(n_{\text{FAT}}\) varies, and Figure 3d shows performance change when \(n_{\text{Paragraph}}\) is changed while other parameters are constant. The best F-measure of 0.294 is achieved for \(m_{\text{FAT}} = 9\), \(m_{\text{Paragraph}} = 2\), \(n_{\text{Paragraph}} = 15\) and \(n_{\text{FAT}} = 75\).

When \(m_{\text{Paragraph}}\) varies, the change in F-measure is not as significant as when \(m_{\text{FAT}}\) varies. In addition, recall is almost constant for \(m_{\text{FAT}} > 2\). This shows that considering more than two GO terms for each sentence in FAT sections does not help us much and can only decrease the precision. On the other hand, considering only one top GO term for the first sentence of each paragraph gives the maximum boost to the recall.

Results comparison

Having the tuned parameters, we compare the performance of the proposed intersection approach with alternative systems (without intersection algorithm or limit on section types). In addition, we compare the contribution of the first and the last sentences of paragraphs. Table 3 shows the performance of different settings. The first
The experiment tests how much the intersection approach improves the results in comparison to just finding semantic similarity of each sentence. The first four rows in Table 3 do not use intersection and simply use the most similar GO term to each sentence. The last five rows in Table 3 use the intersection method. The best recall (0.518) is achieved by not using intersection and not limiting scope to any specific part of the document; however, the precision is low.

Figure 2. This flowchart shows the process of finding GO terms for each gene in a given document by an example. The example sentence category is ‘front_2’ (FAT sections). With the exception of the value for n and m parameters, the process is the same as FAT for sentences in paragraphs.

Figure 3. (a) Top-left diagram depicts precision, recall and F-measure change in respect to mFAT (‘Front’, ‘Abstract’ and ‘Title’) changes when other parameters have constant values (mParagraph = 1, nFAT = 100, nParagraph = 15). (b) Top-right diagram shows the change of performance based on changes of mParagraph when mFAT = 9, nFAT = 100, nParagraph = 15. (c) Bottom-left diagram shows the change of performance when nFAT varies and mFAT = 3, mParagraph = 1, nParagraph = 15. (d) Bottom-right diagram shows the change of performance when nParagraph varies and mFAT = 3, mParagraph = 1, nFAT = 100.
Limiting the scope to paragraph and FAT improved the precision slightly (+0.009) and decreased recall (−0.020). Similarly, including only Paragraph section improved precision and reduced recall a little (+0.010 precision, −0.025 recall). When only the FAT section is included, the precision increased significantly and recall also dropped sharply (+0.199 precision, −0.246 recall). This yields a higher F-measure than including paragraph or all sections. In short, when we limit the scope, the precision increases and recall decreases. We see the same pattern with intersection approach, but precision remains high in comparison with no-intersection approach. When we compare intersection and no-intersection approaches including all sections (Table 3, row 1 and row 5), it shows that intersection reduces recall by 0.213 but increases the precision by 0.186. In another experiment, we found that limiting search to first sentence of paragraph sections can improve the precision significantly. The last four rows of Table 3 compare the performance when different parts of the paragraph are included; they show that including the first sentence yields the best F-measure and precision.

In Table 4, we compared four settings for creating semantic vectors: (i) using only the GO terms, (ii) using GO term and definition, (iii) using GO term and synonym and (iv) using GO term, definition and synonym. For all experiments in this table, FAT and Paragraph (only first sentence) sections are considered.

### Table 3. This table shows performance of different settings on dev-set

<table>
<thead>
<tr>
<th>Setting</th>
<th>Precision</th>
<th>Recall</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>No intersection/All sections included</td>
<td>0.082</td>
<td>0.518</td>
<td>0.141</td>
</tr>
<tr>
<td>No intersection/Paragraph + FAT</td>
<td>0.091</td>
<td>0.498</td>
<td>0.155</td>
</tr>
<tr>
<td>No intersection/Paragraph</td>
<td>0.092</td>
<td>0.493</td>
<td>0.155</td>
</tr>
<tr>
<td>No intersection/FAT</td>
<td>0.281</td>
<td>0.272</td>
<td>0.276</td>
</tr>
<tr>
<td>Intersection/All section</td>
<td>0.268</td>
<td>0.305</td>
<td>0.285</td>
</tr>
<tr>
<td>Intersection/Paragraph last sentence + FAT</td>
<td>0.346</td>
<td>0.245</td>
<td>0.287</td>
</tr>
<tr>
<td>Intersection/Paragraph all sentences + FAT</td>
<td>0.316</td>
<td>0.278</td>
<td>0.296</td>
</tr>
<tr>
<td>Intersection/Paragraph last and first sentences + FAT</td>
<td>0.348</td>
<td>0.261</td>
<td>0.299</td>
</tr>
<tr>
<td>Intersection/Paragraph first sentence + FAT</td>
<td>0.366</td>
<td>0.252</td>
<td>0.298</td>
</tr>
</tbody>
</table>

For intersection approach, the tuning parameter values are $m_{FAT}=9$, $m_{Paragraph}=2$, $n_{Paragraph}=15$ and $n_{FAT}=75$. Random index algorithm random function’s seed was fixed to ‘1234’.

### Table 4. Four settings for creating semantic vectors are compared in this table: (i) using only the GO terms, (ii) using GO term and definition, (iii) using GO term and synonym and (iv) using GO term, definition and synonym. For all experiments in this table, FAT and Paragraph (only first sentence) sections are considered.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Precision</th>
<th>Recall</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create vectors with GO terms only</td>
<td>0.366</td>
<td>0.252</td>
<td>0.298</td>
</tr>
<tr>
<td>Create vectors with GO terms + definitions</td>
<td>0.247</td>
<td>0.229</td>
<td>0.238</td>
</tr>
<tr>
<td>Create vectors with GO terms + definitions + synonyms</td>
<td>0.227</td>
<td>0.196</td>
<td>0.210</td>
</tr>
<tr>
<td>Create vectors with GO terms + synonym</td>
<td>0.197</td>
<td>0.189</td>
<td>0.193</td>
</tr>
</tbody>
</table>

**Conclusion**

We proposed an unsupervised approach to extract gene functions from documents. The proposed approach only uses GO terms’ names for creating semantic vectors. We tried using GO terms description, but it does not help. Using a more fine-tuned vocabulary set for each GO term may result in more accurate vectors and may increase the performance of this method. In addition, using term–term semantic similarity for expanding sentence terms can be evaluated. We used annotations for finding the important passage types, evaluating the method and finding the best settings for the parameters. The main advantage of using unsupervised open-IE technique is that it can easily be generalized and applied to similar relation extraction problems. The results from this method can be used as a baseline for supervised systems. In the future, we plan to combine this approach with supervised techniques. The source code and outputs of each experiment are available in https://code.google.com/p/rainbow-nlp/.

**Funding**

Research reported in this publication was supported by the National Library of Medicine under Award Number R01LM011176.
References

   Database, 2012, bas043.
   Database, 2011, baq036.
   Bethesda, USA, pp. 119–127.
   BMC Bioinformatics, 6 (Suppl. 1), S16.
   BMC Bioinformatics, 6 (Suppl. 1), S21.
    BMC Bioinformatics, 6 (Suppl. 1), S18.
    Database, 2013, bat064.
    pp. 294–311.
    LREC, 2335–2338.