Information Retrieval and Information Extraction in TREC Genomics 2007

Antonio Jimeno
yepes@ebi.ac.uk

Piotr Pezik
pezik@ebi.ac.uk

Dietrich Rebholz-Schuhmann
rebholz@ebi.ac.uk
European Bioinformatics Institute,
Wellcome Trust Genome Campus,
Hinxton, Cambridge, CB10 1SD, UK

Abstract

In TREC Genomics a question/answering task has been proposed. A set of questions with a specific entity of interest is proposed and a set of passages from a collection of full text documents has to be selected from the document collection provided. We have used a two step approach: the first one is recall-oriented retrieval, and the second is an information extraction system that is intended to provide higher precision. We rely on well known techniques like query expansion and resources like MeSH and UMLS. The information extraction techniques are part of the infrastructure of the Text Mining Group at European Bioinformatics Institute.

Using standard information retrieval techniques has been found more beneficial than using more complex processing. Having analyzed the results we find that the performance of query expansion varies for different topics. There are several reasons. Terminological resources may contain ambiguous synonyms or synonyms whose textual usage patterns differ from the usage of the original query terms. On the whole our performance was similar to the mean results from the three performance measures.

1 Introduction

In TREC Genomics a question/answering task has been proposed. A set of questions with a specific entity of interest is proposed and a set of passages from a collection of full text documents has to be selected from the document collection provided. Query/answering systems have been evaluated using different strategies. In TREC Genomics the strategy is closer to a retrieval system since we have been asked to retrieve specific passages from the document collection. The specific answer to the question is not required to be extracted from the spans. We used a two step approach: the first step is a recall oriented retrieval and the second is an information extraction approach that is intended to increase the precision of the retrieved set. In the following section we introduce the different techniques used, then we present the results and an analysis of the different factors in our system and finally we draw some conclusions.

2 Description

2.1 Indexing

The documents are provided by Highwire Press [1] and are in HTML format. The documents have been divided into spans of text. These spans are identified by the *p* tag. Since there are several sections in the documents that are not interesting (e.g. references or authors list) or passages that are too short to contain any interesting information; we have filtered spans based on regular expressions and spans under 300 characters. The remaining spans are indexed.

Our index is based on Lucene [2]. We have proposed an improvement on the scoring function of Lucene [3] based on Singhal et al.[4] because we realized that Lucene gives more relevance to very frequent terms in the documents. This lead to the undesired effect that some spans where ranked higher because they contained frequently the occurrence of one specific term from the topic; as in the case of *cancer*. We introduced modifications to the *tf* (term frequency) function and the *lengthNorm* function in the DefaultSimilarity class of Lucene. The *slope* and *pivot* are 0.2 and 300 repectively. The other variables like $freq$ and $numTerms$ are provided by Lucene.

$$tf = 1.0 + log(freq) \tag{1}$$

$$lengthNorm = \frac{1}{(1.0 - slope) * pivot + slope * numTerms} \tag{2}$$

Even though the integration of Lucene into an IT solution is easy, the possible improvements to the score function are limited by the information made available to the functions in the Similarity abstract class. For instance, if we implement or override the tf function the only information is the term frequency. It is not possible to know any other information from the document or any other statistic from the Lucene index. An assessment of the default similarity function provided by Lucene and the modified scoring function can be found in the Results section. The spans have been split into word tokens and these tokens are converted into their singular form and filtered using standard stop word list.

[1] http://www.highwire.org
[2] http://lucene.apache.org
[3] http://lucene.apache.org/java/2_2_0/api/org/apache/lucene/search/Similarity.html

Entity Type	Source
ANTIBODIES	MeSH
BIOLOGICAL SUBSTANCES	Not considered
CELL OR TISSUE TYPES	MeSH
DISEASES	MeSH
DRUGS	DrugBank
GENES	SwissProt
MOLECULAR FUNCTIONS	GO
MUTATIONS	MeSH
PATHWAYS	Not considered
PROTEINS	SwissProt
STRAINS	Not considered
SIGNS OR SYMPTOMS	MeSH
TOXICITIES	UMLS
TUMOR TYPES	MeSH

Table 1: Entity types and knowledge source selected

2.2 Retrieval

In our approach we combine information retrieval and information extraction. In the retrieval part we used an entity recognizer to link the topic with the query expansion procedure. The recognized entity is identified in a datasource and is used to expand the topic with the synonyms. The retrieved spans are postprocessed with an entity recognizer to identify the entities. These entities are matched with the entities in the topic and a boosting factor depending on the matching with the topic is applied.

2.2.1 Entity Recognizer

The entity recognizer allowed us to identity specific entities occurring in text from a diverse set of entity types. The output of the recognizer is used by the query expansion mechanism and by the boosting of the documents. Table 1 shows that although we considered several terminological resources we have not been able to cover the full range of entity types due to time constraints.

The set of terms provided by the different sources has been processed to avoid different types of ambiguities. We have removed terms from a stop word list, terms with less than two characters or only numbers and very common English terms identified from the Brown Corpus. This processing does not require domain knowledge and removes very ambiguous cases.

We have built the named entity recognizers based on a dictionary look-up approach[4] that offers quite flexibility since we only require the terminology for each semantic type. Ambiguity between the entity types is solved based by prioritizing terms[1].

[4]http://monqjfa.berlios.de

2.2.2 Query Expansion

In order to increase the recall of the set of retrieved passages, we have experimented with three different query expansion techniques. The lexical resource used for this purpose was a term repository developed as part of the Bootstrep project. It currently contains terms denoting genes, protein, chemicals, species as well as other semantic types imported from several bio-ontologies. Some initial filtering and normalisation of terms was applied on the repository[5][6]. We compare the results obtained with the different query expansion techniques and their combinations in the Results section.

Staightforward Expansion The first technique is based on simply fetching synonymous terms from the term repository and translating them into sets of variants joined with the boolean operator OR. The synonymy relationship between the query terms and variants found in the repository is taken at face value. In other words, if a variant is recognized as a synonym of a query term in the repository, it is appended to the query term with an OR operator.

Filters In the second technique, we apply some restrictions on certain types of terms that can be used to expand queries. In particular we filter out some gene and protein names, largely due to the fact that these terms tend to be polysemous. The following restrictions were applied:

1. Very short and potentially polysemous or irrelevant gene/protein names matching patterns such as "[a-zA-Z0-9]{1,2}-[0-9]{1,2}" were discarded altogether.
2. Only synonyms which were likely to be expanded or abbreviated forms of original query terms were selected. This was decided on the basis of string similarity based on [3]. Thus, arbitrary synonyms which are more likely to coincide with terms with other meanings were not included.

Term Boosting The third technique is meant to depress the significance of big synsets based on the intuition that a large set of synonyms for one of the query terms might bias the whole query towards that term. Thus, we have attached a boosting factor to each term which we obtained with the following formula:

$$\frac{1}{(log(n) * alpha) + 1} \tag{3}$$

where n is the number of synonyms obtained for a given topic term and $alpha = 0.5$. Thus, if a protein name such as 'PSD-95' gets expanded into the following set of potentially synonymous sets:

[5]http://www.bootstrep.org
[6]ftp://ftp.ebi.ac.uk/pub/software/textmining/bootstrep/termrepository/20092007

(Dlg4 OR "Discs large homolog 4" OR Sap90 OR Psd95 OR PSD95 OR "Synapse-associated protein 90" OR DLG4 OR "Postsynaptic density protein 95" OR "SYNAPSE-ASSOCIATED PROTEIN 90" OR dlg4 OR "postsynaptic density protein 95" OR "discs large homolog 4" OR PSD-95 OR SAP90 OR Dlgh4)0.42

The set of expanded synonyms gets a relatively low boosting factor of 0.42, as it might contain polysemous terms which could affect the precision of the results retrieved.

2.2.3 Document Boosting

The document retrieval is expected to provide a ranked list of documents with high recall. Due to the term ambiguity some documents are ranked higher even if the words are not relevant to the topic. This has been covered partially in the query expansion, where some abbreviations for proteins have been correctly identified and expanded to the long form.

We propose to analyze the documents and boost the passages that make an explicit reference to the entities identified in the topic. The first 3000 documents are retrieved and annotated. Then the matching entities in the topic and the document are counted and this count ($boosting\_factor$) is combined with the score provided by the retrieval system using a linear combination.

$$span\_score = \alpha * lucene\_score + \beta * boosting\_factor \qquad (4)$$

The factors $\alpha = 1.0$ and $\beta = 0.2$ have been tuned using 2006 TREC Genomics Gold Standard.

2.3 Minimum Span locator

The spans retrieved may contain information irrelevant to topic. To identify the interesting pieces of text in a given span we propose to look for the sentences that are more similar to the topic and keep them. We use a bag-of-words representation of the topic and the span sentences. Then the first sentence for which the similarity was higher from a given threshold is collected. The same technique is used to identify the last sentence that may contain interesting results. The approach is quite conservative since we want to avoid removing interesting parts of the text. The similarity is based on the cosine similarity. The values of the vectors for the sentences and the topic are based on the term frequency and the inverted document frequency obtained from the index built for the spans. The threshold is estimated based on the 2006 TREC Genomics Gold Standard.

$$cos(v_1, v_2) = \frac{v_1 \cdot v_2}{|v_1| \times |v_2|} \qquad (5)$$

RUN	DOCUMENT	ASPECT	PASSAGE2
Maximum	.3286	.2631	.1148
Median	.1897	.1311	.0377
Mean	.1862	.1326	.0398
EBI1Lucene	.1799	.1513	.0404
EBI2Fusion	.1768	.1470	.0401
EBI3Boosting	.1522	.1247	.0339
Lucene	.1634	.1340	.0386

Table 2: Average DOCUMENT, ASPECT and PASSAGE2 for the Maximum, Median, Mean and our official runs

3 Results

We submitted three runs. EBI1Lucene used the modified Lucene index and the minimum span locator (no query expansion or boosting has been applied). EBI3Boosting used the modified Lucene index with query expansion, boosting and the minimum span locator. EBI2Fusion combines EBI1Lucene and EBI3Boosting by doing the sum of the scores for each span, reranking with the new score and taking the first 1000 spans. In table 2 we see the performance of each of our three runs and the maximum, mean and median of the automatic runs in the TREC Genomics.

EBI1Lucene provided the best performance for the three runs that we have submitted. Our runs have a performance comparable with the average of the other participants in TREC. In table 3 we can compare the different approaches with the maximum and median for each one of the individual topics from the automatic runs. As we can see for some topics EBI3Boosting outperformed EBI1Lucene but globally the performance is either the same or worst. In the following sections we will identify the issues for which query expansion and boosting did not have the expected performance.

We have proposed some improvements on Lucene's Similarity class (ref. Indexing section). On table 2 we can find the performance of the standard Lucene and our modified version (EBI1Lucene). The modified version outperforms the standard Lucene in almost all the topics.

We have evaluated the minimum span Locator with the PASSAGE2 measure and EBI11Lucene run with and without the minimum span locator. We found and increase of 7.23% in the PASSAGE2 measure. In most of the cases our approach improved the performance based on just delivering the span without any further processing. On the other hand the PASSAGE2 measure is dependent on the span ranking. This means that PASSAGE2 has a dependency on the DOCUMENT measure and it is difficult to compare the behaviour of the different algorithms provided by the participants.

Table 4 summarizes the results obtained for the abovementioned query expansion techniques and their combinations. Filtering out potentially polysemous gene and protein names without term boosting gave the best results, although

Topic	Max	Median	EBI1Lucene	EBI2Fusion	EBI3Boosting
200	.449/.360/.292	.253/.097/.015	.1991/.1332/.0412	.2944/.1788/.0558	.3366/.1873/.0624
201	.612/.324/.227	.222/.063/.016	.3615/.1728/.0315	.2014/.0594/.0116	.0398/.0062/.0015
202	.131/.129/.017	.024/.002/.001	.0182/.0043/.0006	.0181/.0047/.0008	.0181/.0047/.0008
203	.648/.54/.312	.451/.161/.013	.4784/.3482/.0430	.5260/.3606/.0471	.5143/.3506/.0483
204	.674/.664/.414	.444/.318/.02	.2926/.3177/.0089	.2907/.3517/.0088	.2903/.3553/.0088
205	.269/.115/.016	.067/.03/.004	.1408/.0684/.0145	.1424/.0695/.0145	.1432/.0700/.0147
206	.644/.602/.106	.424/.102/.039	.3796/.1658/.0423	.3954/.1692/.0416	.3954/.1692/.0413
207	.261/.075/.001	.057/.006/0	.0312/.0111/.0008	.0308/.0103/.0008	.0300/.0101/.0007
208	.449/.203/.086	.285/.031/.016	.1249/.0221/.0049	.1299/.0223/.0037	.1295/.0222/.0037
209	.508/.641/.562	.209/.222/.127	.0258/.0461/.0286	.0258/.0462/.0277	.0258/.0463/.0276
210	.246/.184/.057	.08/.032/.008	.1083/.0323/.0098	.1095/.0325/.0084	.1077/.0317/.0084
211	.574/.141/.08	.322/.039/.011	.3631/.0770/.0128	.4110/.0931/.0139	.3677/.0693/.0120
212	.47/.472/.278	.233/.277/.098	.3030/.4071/.1112	.3039/.4036/.1111	.3048/.4039/.1112
213	.698/.673/.179	.432/.198/.077	.3715/.4013/.0884	.3712/.4001/.0892	.3709/.4001/.0897
214	.742/.709/.221	.325/.335/.077	.3361/.5431/.0976	.3254/.5224/.0924	.2960/.4864/.0770
215	.504/.332/.159	.322/.118/.051	.4191/.2679/.1266	.3283/.2345/.0933	.2265/.1651/.0610
216	.156/.192/.014	.046/.015/.002	.0685/.0550/.0029	.0578/.0466/.0026	.0101/.0075/.0005
217	.061/.046/.023	.008/.003/.001	.0163/.0086/.0026	.0123/.0066/.0020	.0016/.0007/.0003
218	.366/.24/.181	.241/.096/.033	.2036/.1490/.0783	.2047/.1494/.0833	.2046/.1491/.0833
219	.204/.324/.06	.014/.005/.001	.0686/.0636/.0167	.0070/.0039/.0003	.0685/.0634/.0369
220	1/1/.549	.27/.312/.066	.2415/.2381/.0134	.1848/.1508/.0025	.0058/.0034/.0001
221	.681/.521/.318	.509/.272/.086	.5675/.4169/.1821	.5704/.4301/.2213	.3626/.3755/.1462
222	.338/.338/.12	.054/.124/.025	.0477/.1746/.0825	.0477/.1746/.0825	0/0/0
223	.315/.157/.052	.146/.079/.005	.0488/.0074/.0009	.0491/.0147/.0016	.0490/.0147/.0016
224	.673/1/.724	.003/.002/0	.0005/.0011/.0001	0/0/0	0/0/0
225	.333/.333/.084	.018/.006/.001	0/0/0	0/0/0	0/0/0
226	.754/.68/.156	.215/.147/.013	.0984/.2422/.0078	.1070/.2437/.0076	.1113/.2502/.0078
227	.366/.416/.161	.187/.095/.032	.2820/.1824/.1156	.2862/.1889/.1166	.2855/.1934/.1140
228	.099/.216/.013	.01/.004/.001	.0140/.0114/.0006	.0143/.0102/.0007	.0141/.0100/.0007
229	.643/.804/.247	.278/.293/.044	.2074/.2568/.0343	.3448/.2932/.0647	.3844/.3292/.0759
230	.367/.466/.249	.213/.187/.091	.2264/.2747/.1951	.2129/.2705/.1914	.1969/.1527/.1716
231	.277/.248/.047	.059/.024/.001	.0465/.0289/.0003	.0322/.0528/.0004	.0042/.0016/0
232	.28/.211/.028	.081/.019/.002	.1640/.0945/.0086	.1780/.1073/.0083	.1412/.0754/.0056
233	.215/1/.065	.04/.029/.001	.0745/.1429/.0084	.0564/.125/.0062	.0294/.0714/.0028
234	.221/.24/.103	.077/.073/.01	.0054/.0048/.0003	.0054/.0048/.0003	.0054/.0048/.0003
235	.442/.364/.34	.117/.064/.012	.1422/.0764/.0405	.0885/.0612/.0310	.0063/.0079/.0021

Table 3: DOCUMENT, ASPECT and PASSAGE2 for the Maximum, Median and our official runs per topic

Expansion	DOCUMENT	ASPECT	PASSAGE2
EBI1Lucene (no expansion)	.1799	.1513	.0404
No filtering, no term boosting	.1455	.1162	.0293
Filtering, no term boosting	.1511	.1242	.0337
No filtering, term boosting	.1352	.1130	.0290
Filtering, term boosting	.1453	.1200	.0317

Table 4: Average DOCUMENT, ASPECT and PASSAGE2 for the EBI1Lucene and the different query expansions

on average they were still worse than the baseline modified Lucene score. Figure 1 shows the biggest increments and decrements in the document per query score after applying query expansion. The biggest depression of performance was noted for topic 201. This could be due to the fact that one of the query terms *cancer* was expanded into several apparent synonyms such as *tumor* or *neoplasm*. Although their denotative meaning is similar from the point of view of an abstracted lexical resource (MESH in this case), in real textual usage cancer, tumor and neoplasm may have a complementary distribution. As an example, the term *tumor* could denote any increase of the size of a tissue or an organ, such as the enlargement of a gland. *Neoplasm* typically refers to the generation of new tissue, whereas *cancer* has the additional characteristics of invading surrounding tissues, dislocating and forming neoplasms elsewhere (metastasis). Because all these different senses are bundled together under one MESH entry and thus the expansion of the term *cancer* leads to a very dispersed query. Topic 200 is an example of a successful expansion. The term *lupus* was expanded with its rarer full form *lupus vulgaris*, whose exact occurrences in some of the passages produced a better idf score than could have been obtained if only the occurrences of the more frequent short form *lupus* were considered.

We have applied boosting in EBI3Boosting after query expansion. As we can see in figure 1 boosting increased slightly the DOCUMENT measure for a large number of topics but decreased the performance from some of the topics like 204. In this topic, the term *neurosteroids* has not been recognized so documents that were more specific to *nervous system* and *brain* have been given more relevance. The effectiveness of the method requires high precision and recall information extraction and a high coverage of the biomedical terminology. The variety of topics does not allow further tuning as proposed in [2] where a template-dependent selection of entities provided higher effectiveness.

4 Conclusion

Modification of Lucene has provided our best run, although the Lucene scoring function can be further improved. These improvements are limited by the way the Lucene package is designed. We expected this run to be the baseline for our other two runs. We have found that query expansion and boosting had

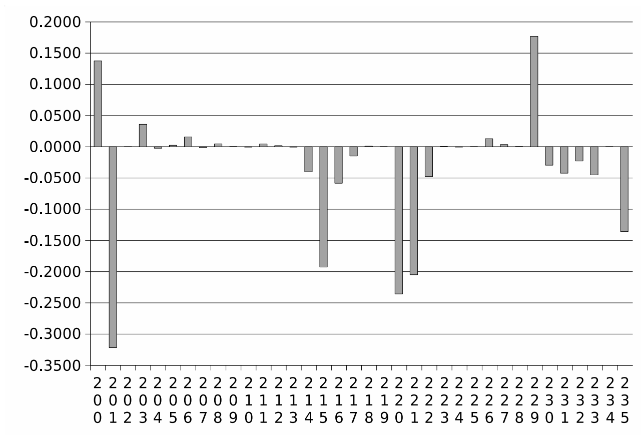

Figure 1: Increment of DOCUMENT per query using the query expansion applying filtering and no term boosting and document boosting versus the modified Lucene

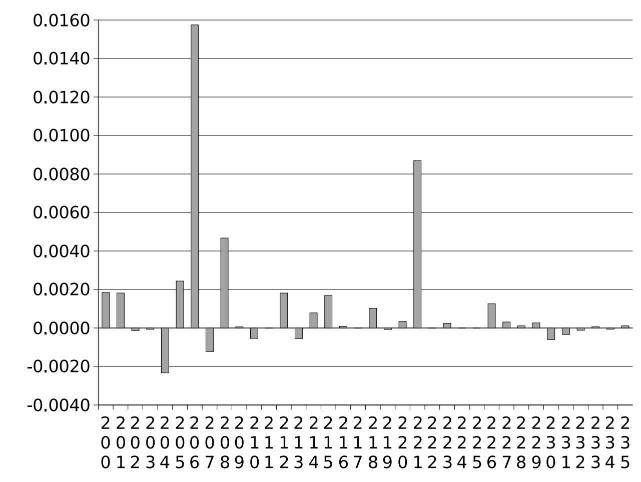

Figure 2: Increment of DOCUMENT per query using document boosting versus the query expansion applying filtering and no term boosting

very different behaviour for the different topics and this may indicate that it is difficult to use a unified technique for different topics [5]. In addition, expanded terms can have a different distribution of usage as in the case of MeSH synonyms for *cancer*. The document boosting suffered from the coverage of the entity recognizer since in some cases a relevant entity was not detected and the spans containing the other recognized entities were prefered. The fusion run (EBI2Fusion) did not provide the desired effect and this may be due to the fact that the two runs used for the fusion were based on the same retrieval engine and the variations like query expansion and boosting applied in one of the runs did not provide a positive effect thus decreasing the performance of the combined run.

5 Acknowledgements

This work was funded by the Network of Excellence "Semantic Interoperability and Data Mining in Biomedicine" (NoE 507505) and by the EC project BOOTStrep FP6-028099 (www.bootstrep.org).

References

[1] Dietrich Rebholz-Schuhmann, Harald Kirsch, Sylvain Gaudan, Miguel Arregui, and Goran Nenadic. Annotation and Disambiguation of Semantic Types in Biomedical Text: a Cascaded Approach to Named Entity Recognition. *Workshop on Multi-Dimensional Markup in NLP, EACL. Trente, Italy*, 2006.

[2] Patrick Ruch, Antonio Jimeno Yepes, Frederic Ehrler, Julien Gobeill, and Imad Tbahriti. Report on the TREC 2006 Experiment: Genomics Track. *the Fifteenth Text Retrieval Conference*, 2006.

[3] A.S. Schwartz and M.A. Hearst. A simple algorithm for identifying abbreviation definitions in biomedical text. *Pacific Symposium on Biocomputing*, 8:451–462, 2003.

[4] Amit Singhal, Chris Buckley, and Mandar Mitra. Pivoted document length normalization. In *SIGIR '96: Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 21–29, New York, NY, USA, 1996. ACM Press.

[5] Ellen M. Voorhees. Query expansion using lexical-semantic relations. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 61–69. Springer-Verlag New York, Inc., 1994.