

Identifying Relevant Full-text Articles for Database Curation

Chih Lee, Wen-Juan Hou and Hsin-Hsi Chen

*Department of Computer Science and Information Engineering,
National Taiwan University, Taipei, Taiwan, 106*

E-mail: {clee, wjhou }@nlg.csie.ntu.edu.tw; hhchen@csie.ntu.edu.tw

Abstract

In this paper, we illustrated our approach to identifying curatable articles from a large pool. Our system currently considered three parts of an article as three individual representations of the article and utilized two domain-specific resources to reveal the deep knowledge contained in the article, generating more representations of the article. Cross-validation was employed to find the best combination of representations and an SVM classifier was trained out of this combination. The cross-validation results and results of the official runs were listed, showing overall high performance.

1 Introduction

Organism Database plays a crucial role in genomic and proteomic research. It stores the up-to-date profile of each gene of the species interested. To provide biomedical scientists with easy access to complete and accurate information, curators have to constantly update databases with new information. Published literature written in natural languages has long been the main source of information because of its high accuracy. With the rapidly growing rate of publication, it is impossible for database curators to read every published article. However, since current fully-automated curation systems have not met the strict requirement of high accuracy and recall, database curators still have to read some (if not all) of the articles sent to them. Therefore, it will be very helpful if a triage system is able to correctly identify the curatable or relevant articles in a large pool.

2 Architecture Overview

Figure 1 shows the overall architecture of our system for the categorization task. For each full-text training article, we first preprocessed the article and extracted several parts from it. Each of the extracted parts was considered a representation of this article. In this task, we considered three parts of an article, which are 1) title and abstract, 2) MeSH terms assigned to this article and 3) figure and table captions. With the help of domain-specific knowledge, we processed the three parts and obtained more representations of an article, while the original three representations were kept. The three original representations are denoted as **Abstract**, **MeSH** and **Caption** in the rest of this paper. In the model selection phase, we performed feature ranking on each representation of an article and employed cross-validation to decide the number of features to be kept. Moreover, we used cross-validation to obtain the best combination of all the representations. Finally, a support vector machine (SVM) (Vapnik, 1995; Hsu *et al.*, 2003) classifier was obtained.

3 Methods

3.1 Document Preprocessing

In the preprocessing phase, we performed acronym expansion on the articles, removed the remaining tags from the articles and extracted three parts of interest from each article. Abbreviations are often used to replace long terms in writing articles, but it is possible that several long terms share the same short form, especially for gene/protein names. To avoid ambiguity and enhance clarity, the acronym expansion operation replaces every tagged abbreviation with its long form followed by itself in a pair of parentheses. An example of this operation is shown in Figure 2, where a tagged abbreviation “IP₃” will be replaced with

“inositol trisphosphate (IP₃)”.

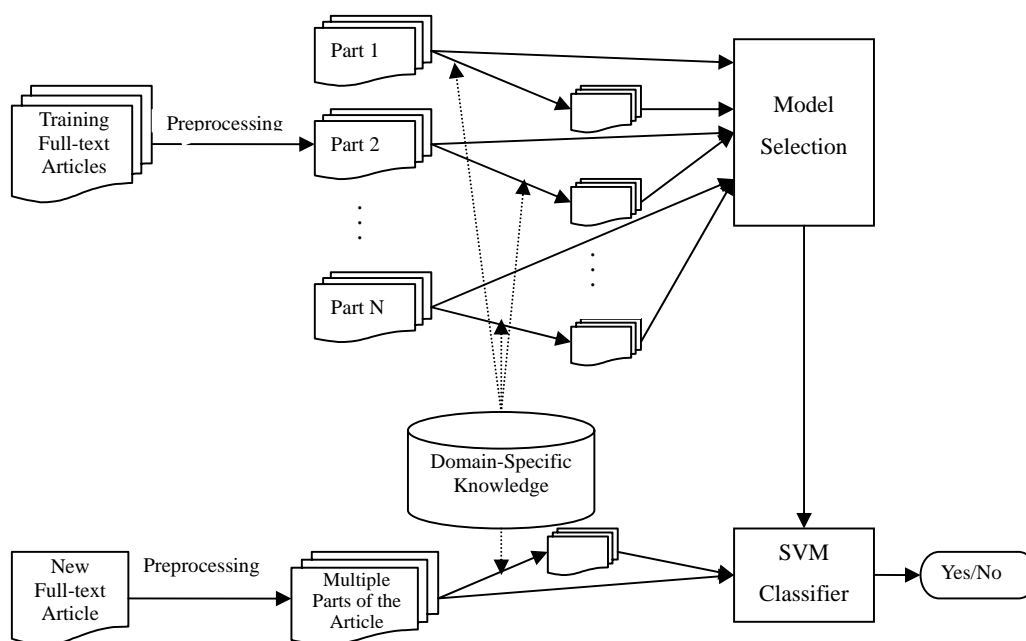


Figure 1: System architecture.

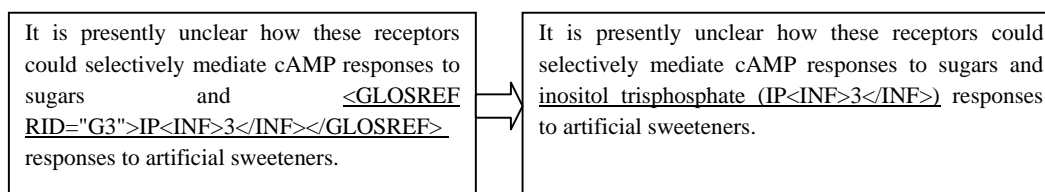


Figure 2: An example of acronym expansion operation.

3.2 Tapping Domain-Specific Knowledge

An article is simply a sequence of tokens separated by delimiters, which is meaningless to machines. With the help of domain-specific knowledge, we can extract the deep knowledge from a piece of text written in natural language. For example, with a gene name dictionary, we can identify the gene names contained in an article. Moreover, by further consulting organism databases, we can get the properties of the genes occurred in the article. Two domain-specific resources were exploited in this task. One is the Unified Medical Language System (UMLS) (Humphreys *et al.*, 1998) and the other is a list of tumor names obtained from Mouse Tumor Biology Database (MTB)¹.

UMLS contains a very large dictionary of biomedical terms – the UMLS Metathesaurus and defines a hierarchy of semantic types – the UMLS Semantic Network. Each concept in the Metathesaurus contains a set of strings, which are variants of each other, and belongs to one or more semantic types in the Semantic Network. Therefore, given a string, we can obtain a set of semantic types to which it belongs. For each part of an article extracted during preprocessing, we obtained another representation of the article by gathering the semantic types found in the part of the article. Consequently, we got three more

¹ <http://tumor.informatics.jax.org/mtbwi/tumorSearch.do>

different representations of an article after this step. They are denoted as **AbstractSEM**, **MeSHSEM** and **CaptionSEM**.

We used the list of tumor names only on the Tumor subtask. We first tokenized all the tumor names and stemmed each unique token, where the tokenization operation considers a sequence of successive alphanumeric characters as a token. With the resulting list of unique stemmed tokens, we used it as a filter to remove the tokens not in the list from the **Abstract** and **Caption** representations, which produced representations **AbstractTM** and **CaptionTM**.

3.3 Model Selection

As mentioned above, we have several representations for an article. In this section, we explain how feature selection was done for each representation and how the best combination of the representations of an article was obtained. In the following paragraphs, the word “token” refers to different concepts, depending on the representations of an article. A token is a stemmed sequence of consecutive alphanumeric characters for the **Abstract**, **MeSH**, **Caption**, **AbstractTM** and **CaptionTM** representations. For the **AbstractSEM**, **MeSHSEM** and **CaptionSEM** representations, a token is a semantic type.

For each representation, we first ranked all the tokens in the training documents via the chi-square test of independence. Assuming the ranking perfectly reflects the effectiveness of the tokens in classification, we then decided the number of tokens to be used in classification by 4-fold cross-validation. In cross-validation, we used the well-known bag-of-words model with TF*IDF (term frequency inverse document frequency) weighting. Each feature vector was normalized to a unit vector after weighting. We adopted SVMs as our classification system and set C_+ to $u_r * C_-$ because of the relatively small number of positive examples, where C_+ and C_- are the penalty constants on positive and negative examples in SVMs. After cross-validation, we obtained the optimal number of tokens and the corresponding SVM parameters C_- and γ , a parameter in the radial basis kernel. In the following paragraphs, **Abstract30** denotes the **Abstract** representation with top-30 tokens, **CaptionSEM10** denotes **CaptionSEM** with top-10 tokens, and so forth.

After feature selection was done for each representation, we tried to find the best combination of the representations by a simple algorithm, where combining two or more representations was achieved by simply concatenating the feature vectors. Therefore, under a combined model of N representations, each article is represented as an N -unit long feature vector. The algorithm is described below.

Given the candidate representations with selected features, e.g. **Abstract10**, **Caption10** and **Mesh30**, we start with an initial set containing some or zero representation. For each iteration, we add one representation to the set by picking the one that enhances the cross-validation performance the most. The iteration stops when we have exhausted all the representations or adding more representation to the set doesn't improve the cross-validation performance.

In the categorization task, we ran the algorithm twice. We first started with an empty set and obtained the best combination of the basic three representations, e.g. **Abstract10**, **Caption10** and **Mesh30**. Then, starting with this combination, we attempted to incorporate the three semantic representations, e.g. **Abstract30SEM**, **Caption10SEM** and **Mesh30SEM**, and obtained the final combination. Instead of using this algorithm to incorporate the **AbstractTM** and **CaptionTM** representations, we used them to replace their unfiltered counterparts **Abstract** and **Caption** when their cross-validation performance was better.

4 Results and Discussions

Table 1 lists the individual cross-validation result (in NU measure) of each representation for each subtask. For subtask Allele, the **Caption** representation performed the best among the basic representations, while **AbstractSEM** performed the best among the semantic representations. For subtask Expression, we can see that captions in articles play an important role in identifying relevant documents, which agrees with the finding by the winner of KDD CUP 2002 task 1 (Regev *et al.*, 2002).

Similarly, we can infer that MeSH terms are crucial to the GO subtask, which also supports the wide-spread using of MeSH terms by top-performing teams (Dayanik *et al.*, 2004; Fujita, 2004) in TREC Genomics 2004. Looking at the Tumor subtask, we can tell that MeSH terms are important, but after semantic type extraction the **AbstractSEM** representation exhibited relatively high cross-validation performance. Since only 10 features were selected for the **AbstractSEM** representation, using this representation alone may be susceptible to overfitting. Finally, by comparing the performance of the **AbstractTM** and **Abstract** representations, we found the list of tumor names helpful for filtering abstracts.

Table 1: Partial cross-validation results.

	Allele	Expression	GO	Tumor
	# Tokens / NU	# Tokens / NU	# Tokens / NU	# Tokens / NU
Abstract	10 / 0.7707	10 / 0.5586	10 / 0.4411	10 / 0.8055
MeSH	10 / 0.7965	10 / 0.6044	10 / 0.4968	30 / 0.8106
Caption	10 / 0.8179	10 / 0.7192	10 / 0.4091	10 / 0.7644
AbstractSEM	10 / 0.7209	10 / 0.4811	10 / 0.3493	10 / 0.8814
MeSHSEM	10 / 0.6942	10 / 0.4563	10 / 0.4403	10 / 0.7047
CaptionSEM	30 / 0.6789	10 / 0.5433	10 / 0.2551	30 / 0.7160
AbstractTM				30 / 0.8325
CaptionTM				10 / 0.7498

We list the results of our official runs in Table 2. Column “cv NU” shows the cross-validation NU measure, “NU” shows the performance on the test data and column “combination” lists the combination of representations used for each run. In this table, M30 is the abbreviation for the **MeSH30** representation, CS10 represents the **CaptionSEM10** representation, and so on. The combinations for the first 4 runs were obtained by the simple algorithm described in Section 3, while the combination for tNTUMACwj was obtained by substituting **AbstractTM30** for **Abstract30** in the combination for tNTUMAC. The last run tNTUMACasem used only the **AbstractSEM10** representation because its cross-validation performance beat all other combinations for the Tumor subtask.

The combinations of the first 5 runs illustrate that adding other inferior representations to the best one enhanced the performance, which implies that the inferior ones may contain important exclusive information. The cross-validation performance fairly predicted the performance on the test data, except for the last run tNTUMACasem, which relies on only 10 features and is therefore susceptible to overfitting.

Table 2: Results of our official runs.

Run Tag	cv NU	NU	Recall	Precision	F-score	combination
aNTUMAC	0.8717	0.8423	0.9488	0.3439	0.5048	M30+C10+A10+CS10+AS10+MS10
eNTUMAC	0.7691	0.7515	0.8190	0.1593	0.2667	M10+C10+CS10+MS10
gNTUMAC	0.5402	0.5332	0.8803	0.1873	0.3089	M10+C10+MS10
tNTUMAC	0.8742	0.8299	0.9000	0.0526	0.0994	M30+C30+A30+AS10+CS30
tNTUMACwj	0.8764	0.8747	0.9500	0.0518	0.0982	M30+C30+AT30+AS10+CS30
tNTUMACasem	0.8814	0.5699	0.6500	0.0339	0.0645	AS10

5 Concluding Remarks

In this paper, we demonstrated how our system was constructed. Three parts of an article were extracted and each of them was considered a representation of the article. We incorporated two domain-specific resources – UMLS and a list of tumor names. By integrating domain knowledge, we obtained 5 more representations of an article. We performed feature selection on each of the 8 representations to obtain the optimal number of features for each of them. For each subtask, we mainly relied on a simple algorithm to get the best combination of the representations and trained an SVM classifier out of this combination. The partial cross-validation results and the results of our official runs were listed.