SVM-based algorithms for aligning ontologies using literature

by

Wei Xu

LIU-IDA/LITH-EX-A--08/058--SE

2008-12-15
Final Thesis

SVM-based algorithms for aligning ontologies using literature

by

Wei Xu

LIU-IDA/LITH-EX-A--08/058--SE

2008-12-15

Supervisor: Patrick Lambrix, He Tan
Department of Computer Science and Information Science at Linköping University

Examiner: Patrick Lambrix
Department of Computer Science and Information Science at Linköping University
Ontologies is one of the key techniques used in Semantic Web establishment. Nowadays, many ontologies have been developed and it is critical to understand the relationships between the terms of the ontologies, i.e. we need to align the ontologies. This thesis deals with an approach for finding relationships between ontologies using literature by classifying documents related to terms in the ontologies.

In this project the general method from [1] is used, but in the classifier generation part, a brand new classifier based on SVMs algorithm is implemented by LPU and SVM\textsuperscript{light}. We evaluate our approach and compare it to previous approaches.
Abstract:

Ontologies is one of the key techniques used in Semantic Web establishment. Nowadays, many ontologies have been developed and it is critical to understand the relationships between the terms of the ontologies, i.e. we need to align the ontologies. This thesis deals with an approach for finding relationships between ontologies using literature by classifying documents related to terms in the ontologies.

In this project the general method from [1] is used, but in the classifier generation part, a brand new classifier based on SVMs algorithm is implemented by LPU and SVM\textsuperscript{light}. We evaluate our approach and compare it to previous approaches.

**Key words:** Ontology alignment, text classifier, SVM, LPU, SVM\textsuperscript{light}
Acknowledgements

This thesis work was carried out at IISLab of Department of Information and Computer Science (IDA), where I enjoyed a lot within the whole project.

I would like to specially thank my examiner and supervisor Patrick Lambix for the help and guidance during this thesis. His suggestion and timely advice helped me shape this thesis and was invaluable to the completion of this work. I appreciate a lot for his encourage and patience especially. I also want to thank my supervisor He Tan for her helpful discussion and explanation in details.

Another special thank goes to my parents and friends who support me all along in my study and in my life.

The time being there during my project has become a nice memory for me in my life forever.

Linkoping, December 2008

Wei Xu
Contents

Chapter 1 ........................................................................................................................5
Introduction ....................................................................................................................5
  1.1 Motivation: ........................................................................................................5
  1.2 Problem statement: ............................................................................................6
  1.3 Proposed solution: .............................................................................................7
  1.4 Outline: .............................................................................................................7
Chapter 2 ........................................................................................................................8
Background: ...................................................................................................................8
  2.1 Ontology: ..........................................................................................................8
  2.2 Ontology alignment: .........................................................................................9
  2.3 Methods for Ontology Alignment: ..................................................................10
    2.3.1 Ontology Alignment Systems................................................................11
    2.3.2 Ontology Alignment Strategies ...............................................................11
    2.3.3 Document-based Ontology Alignment .................................................12
  2.4 Algorithm and Software..................................................................................13
    2.4.1 Support vector machines (SVMs): ........................................................13
    2.4.2 Software introduction............................................................................14
Chapter 3 ......................................................................................................................16
Analysis, Design and Implementation .........................................................................16
  3.1 Analysis of Algorithm and software ...............................................................16
    3.1.1 Basic Algorithm....................................................................................16
    3.1.2 Document-based ontology alignment with NB Algorithm...................16
    3.1.3 Document-based ontology alignment with SVMs Algorithm ..............17
  3.2 Alignment Algorithm Design .........................................................................19
    3.2.1 Basic Algorithm....................................................................................19
    3.2.2 Alternative: plural and single classification..........................................20
  3.3 Implementation ...............................................................................................20
    3.3.1 Environment Set Up..............................................................................20
    3.3.2 Data Formulation .................................................................................21
    3.3.3 Data preparation and format transformation.........................................22
    3.3.4 Calling external command in Java........................................................25
    3.3.5 Calculation ............................................................................................25
Chapter 4 ......................................................................................................................27
Evaluation ....................................................................................................................27
  4.1 Description of the evaluation ...........................................................................27
  4.2 Evaluation procedure ......................................................................................27
  4.3 Evaluation results .............................................................................................28
    4.3.1 Results analysis............................................................................................30
    4.3.2 Summary......................................................................................................32
Chapter 5 ......................................................................................................................33
<table>
<thead>
<tr>
<th>Conclusion and Future work</th>
<th>33</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1 Conclusion</td>
<td>33</td>
</tr>
<tr>
<td>5.2 Future work</td>
<td>33</td>
</tr>
<tr>
<td>Appendix A</td>
<td>35</td>
</tr>
<tr>
<td>Appendix B</td>
<td>36</td>
</tr>
<tr>
<td>Appendix C</td>
<td>37</td>
</tr>
<tr>
<td>Bibliography</td>
<td>38</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

1.1 Motivation:

The World Wide Web never stops progressing. As we experience an explosion of the amount of information, as the amount of data grows and floods, researchers and users need to do more work to retrieve and reuse the information. In order to represent data in a way that it will be easier for computers to help us with storing, managing and finding information, the Semantic Web is being developed. It provides a common framework that allows data to be shared and reused across applications, enterprises, and community boundaries. [2].

The Semantic Web aims at maintaining better organized, clearly defined, properly managed and easily retrieved information, which should be given well-defined meaning. In a first step this is done by annotating information with ontology terms. Ontologies are suitable for classifying and defining the concepts, relations, domains, classification rules and so forth. Ontologies is one of the key techniques used in Semantic Web establishment.

Currently, a large number of biomedical ontologies have been developed, and databases which contain information about genes, gene sequence information, proteins, gene functions, etc. have been annotated using the terms in these biomedical ontologies. Therefore, it is critical to understand the relationships between the terms of the ontologies. Finding the relationships between terms in different ontologies, we need to align the ontologies. We can also merge two ontologies by using the ontologies and their relationships and create a new ontology. Based on various alignment strategies, a number of systems for aligning ontologies have been developed and many of them include matchers as an essential technique. Matchers are used to recognize semantic relations between ontologies and give results from the alignment systems to assist user finding the relationships.

In the progress of ontologies alignment, the critical technique is to compute similarity based on different strategies. Since there are many optional strategies to choose from, which one is better suited to which occasions is an interesting and challenging task. In
recent years, there is a growing interest in instance-based strategies for ontology alignment. These strategies compute similarities between ontology concepts based on the instances of the concepts. Some of the instances are from literature or instances within the ontologies themselves. In this paper we propose an instance-based strategy using documents from literature. We employ Support vector machines (SVMs) algorithm for generating text classifiers and use these text classifiers to compute similarity. We base our work on previous work [1] where a Naive Bayes algorithm was used for classifier generation. We present the SVMs algorithm and do a comparative study with the results of both.

1.2 Problem statement:

Dealing with the ontology alignment, choosing the right matcher plays a key role to compute good alignment suggestions. Usually, there are several candidate strategies that can be implemented in the similarity calculation. They are based on linguistic matching, structure-based strategies, constraint-based approaches, instance-based strategies, strategies that use auxiliary information or combination of these [a]. Sometimes, there may be more than one strategy used in one learning matcher system. Therefore, based on different strategies and their combinations there come various approaches of alignment which makes that an interesting and challenge topic for research.

With so many strategies, people are trying to find out a particular way which suit certain occasion. One best way of verifying the performance of different method is through comparative studies. After the study of [1] whose strategy is based on the instance-based matching, we found it so interesting and researchable to learn matchers based on instances with different algorithms. In order to verify the flexibility and validity of this strategy, we proposed to apply other algorithms on the same instance data set trying to find out how the algorithm performs and which one is better or more suitable in certain conditions.

We keep the same dataset from the previous work [1], of which the textual content is from some of the life science literatures. Instead of Naive Bayes as the classifier generation algorithm, we introduced SVMs algorithms. In this paper, we address the following problems:

1. How does SVMs work?

2. Can it be applied in biomedical ontology alignment?

3. Why can SVMs perform well for instance-based Strategies?

4. If positive, how to achieve that?
5. The instances we have are only positive and unlabeled (no negative).

6. Which software suits our task best and how to implement?

1.3 Proposed solution:

Once the problem confirmed, the solution to the problem will be how and with what to solve it. The first two questions can be accomplished by reading and researching, while the most work comes from implementation. With the data derived from Web resources, we found out that all the data we have is positive or untargeted, which doesn’t identify the regular classification requirement. Luckily, the software LPU (learning from positive and unlabeled examples) helped us solving with this problem, therefore with all the tools we need, implementation can be carried on.

1.4 Outline:

- **Chapter 1**, Brief introduction of this Master thesis, involving motivation, problem statement, and proposed solution.

- **Chapter 2**, Ontology and Ontology Alignment specific introduction together with alignment methods and SVMs description.

- **Chapter 3**, Brief characterizations of analyses, basic algorithm and each steps of classification process.

- **Chapter 4**, Test evaluation referring to the experimental result in our project is mentioned including the steps, results and analysis.

- **Chapter 5**, In this chapter, we reached a conclusion about the whole project and made a perspective of future work.
Chapter 2

Background:

2.1 Ontology:

Nowadays, over the web, there exists plenty of information which belongs to different domains, subjects, and organizations. In each field, the information is developed independently and they have their own rules for creating and defining terms for exchanging information. However, different terms in different information sources may have the same meaning. The same term can be used in different sources to represent different things. All this is confusing for the users and that is why Ontologies are established as a key technique in the Semantic Web.

The original meaning of the word “ontology” comes from philosophy, where it is defined as the theory of existence in nature. In the computer science domain, there are many different descriptions about ontology. This thesis follows an early definition provided in the field of Artificial Intelligence [3]: An Ontology defines the basic concepts and relations of a domain of interest, as well as the rules for combining these terms and relations. Though the definition is a rough description, it is explicit that Ontology includes both of the defined and inferred terms with rules.

Ontologies can be used for communicating between different users and organizations by sharing common knowledge of terms across platforms, improving the interoperability, maintenance, reliability and documentation between systems.[4] Additionally, ontologies can make the content in information sources well defined and can be presented as an index for retrieving and reusing information. Besides, ontologies can form a repository of information to be queried for other information sources. Apart from those, ontologies can support domain and application-based knowledge respectively as well as validation of data sources. In a word, the benefits of using ontologies can be concluded as: better understanding, more efficiently handling and more easily maintenance. Due to these advantages ontologies as an essential technique has been employed in many fields, for instance, Bioinformatics, some of the Genomics and Semantic Web development.
In recent years, there is a lot of international cooperation for research on Biomedical Ontologies, such as the Gene Ontology (GO) [6], Open Biomedical Ontologies [7] and SNOMED [8]. The biomedical ontologies in this paper are GO ontologies, Signal-Ontology (SigO), Medical Subject Headings (MeSH) and the Anatomical Dictionary for the Adult Mouse (MA).

2.2 Ontology alignment:

With the development of ontologies in many areas with different standards, there exists plenty of overlapping information between different ontologies as shown in Figure 2.1. For example, there are a bunch of terms related to “nose” in both MA and MeSH ontologies. Some of the terms may different from the names or forms they were presented, but same virtually. In order to use ontologies across different platforms, alignment or merging are needed. For instance, companies or organizations may have ontologies based on their own specific requirements. However, often they also need to use public ontologies as most databases are annotated with terms from the public ontologies and they need these to query all these databases.

![Figure 2.1 example of overlapping ontologies](image)

Ontology alignment is the automated resolution of semantic correspondences between the representational elements of heterogeneous systems [9]. It is to define the relationships between concepts or relations from two different source ontologies [10]. However, alignments are not precise [11]. Since each alignment is based on the relations of ontology elements and any particular element alignment will depend on the alignments between other elements. Moreover, element alignments can (and often do) have degrees of confidence associated with them. That is, the aligner cannot say with certainty that any particular alignment is true, only that it is the most probable alignment given other alignments. A general framework that describes most of the current alignment systems is described in paper [12] (see figure 2.2).
Figure 2.2 shows a general framework for ontology alignment. Two ontologies are accepted as input to the system. The system contains five main parts: alignment algorithm, combination/filter, suggestions, suggestion judgment and conflict checker. The alignment algorithm may include more than one matcher with different strategies. Based on various strategies or their combination, a similarity value between terms from the source ontologies is calculated. The results are combined and/or filtered to obtain suggestions which are sent to the user for acceptance or rejection. Finally, all the results, positive or negative, will be sent back and stored in the database, which may influence the computation of future suggestions. In addition, the conflict checker is used to avoid conflicts generated by the alignment relationships. Some systems will also have a merging part that merges the two ontologies using the alignment. We will not discuss merging further in this paper.

2.3 Methods for Ontology Alignment:

The number of methods for ontology alignment is increasing and available nowadays. Although, the existing methods for ontology alignment are designed for various purposes using different techniques, the core technique for many of them is to compare similarities between ontologies with some strategies cooperate with certain algorithm. Via the interpretation of the results, they present some possible set of
2.3.1 Ontology Alignment Systems

The approaches of aligning ontologies have been studied for years and some alignment systems are developed to assist people finding the relationships and facilitate the identification between ontologies. Systems for ontology alignment include PROMPT [13], FOAM [14], CUPID [15], FCA-Merge [16], HCOME-Merge [17], AnchorPrompt [18], GLUE [19], SAMBO [5] and so forth, some of which have the merging function as well. Most of the aligning systems follow the general framework as presented in Figure 2.2. Many alignment systems equipped with several matchers which are used to calculate the similarities between concepts from different ontologies. The core technique within the alignment process is all around the matchers, which include strategies selecting, matchers cooperating and algorithms implementing. In this paper, we concentrate more about the alignment part in the system. We develop an instance-based strategy based on SVMs (Support vector machines) algorithm.

2.3.2 Ontology Alignment Strategies

Regarding the importance of matcher strategies, some works have compared and evaluated the strategies and their performance. Six strategies so far are summarized and adopted in ontology alignment for various purposes. According to the description in [5], six strategies are listed below:

- **Strategies based on linguistic matching**
  
  Linguistic study encompasses the structure and meaning of concepts, which could be helpful in comparison and classification of concepts. For ontology alignment, similarity calculation can found on the textual description of the concepts and relations and definitions [5], for example, the frequency of string appearance counting, semantic and grammar analysis etc.

- **Structure-based strategies**
  
  Usually, two relationships between terms in ontology alignment are used: “is-a” and “part-of”. The existing methods of ontology alignment based on the structures, use the structural environment of concepts to influence the similarity of concepts.

- **Constraint-based approaches**
  
  Besides the methods above, it can also be estimated for concepts or relations
whether they are the same by the properties of themselves. There are ranges and bounds for each relation, which reveal their relations when being compared. It may be effective for some simple comparison. Most of the time, constraint-based approaches can cooperate with other strategies as a combination strategy.

- **Instance-based strategies**

  Over the internet, there exist many data sources where the data entries are annotated with ontology terms. These entries can be collected as instances. Some ontologies also already have instances. In these methods a similarity between concepts is calculated based on their instance.

- **Use of auxiliary information**

  Sometimes, assistant tools may help with alignment as well. Dictionaries, thesauruses, alignment database or other available ontologies are auxiliary information, which can be utilized during the process of ontology alignment.

- **Combining different approaches**

  Ontology alignment is a complex and fuzzy task, patience and seriousness are required. Besides of that, combination of different strategies and approaches are pretty helpful with the similarity calculation and the final results, since there is no strategy suitable for every ontology alignment. Actually, nowadays, many systems of ontology alignment implement more than one strategy.

### 2.3.3 Document-based Ontology Alignment

Document-based ontology alignment is a special case of instance-based strategy. This paper is a continuation of the work presented in [1], where a document-based strategy was proposed. In this paper, we use the corpora of abstracts from four of the biomedical literatures: GO ontologies, Signal-Ontology, Medical Subject Headings (MeSH) and the Anatomical Dictionary for Adult Mouse (MA). As discussed before, the critical technique in ontology alignment is the similarity calculation between terms from different ontologies. We realize that most of the ontologies we got are in the form of documents. The terms for comparing are literal as instance. The text classification happened to be a hot topic nowadays and many approaches are under development. Because of the availability of literal instance from many mature biomedical ontology websites, there is a huge opportunity of aligning ontologies with the method of text-classification. Hence, in this paper, we keep the documents as instances but replace the Naive Bayes with Support vector machines algorithm in the step of classifier generating (detailed description in Chapter 3).

Figure 2.3 presents a general idea of alignment algorithm based on [1].
Step 1: Generate corpora. Based on the documents from the ontologies we want to align, a corpus of different sizes is generated.

Step 2: Generate classifier. For each ontology one of more classifiers is generated based on the corresponding corpus.

Step 3: Classification. The generated classifiers of one ontology are used to classify the documents in the other ontology and vice versa.

Step 4: Similarity calculation. In the end, similarity values between the concepts from different ontologies are computed based on the classification of the documents.

![Figure 2.3: a general view of ontologies alignment](image)

2.4 Algorithm and Software

When problems brought forward, it is clearer what we need to do most. Since matchers are the core technique, strategies and algorithms choosing are the most important part. In this section, the algorithm SVMs we proposed will be introduced briefly together with the background research of the applications we are applying with.

2.4.1 Support vector machines (SVMs):

SVMs (Support vector machines) is a machine learning algorithm used for data classification and regression. It was developed by Vladimir Vapnik and his colleagues in 1995 [20], based on statistical learning theory. From a geometrical perspective, data
can be represented as a set of feature vectors in n-dimensional space. According to the attributes of training vector sets, SVMs splits them into different feature spaces and generates a hyperplane which is a decision function to predict new unclassified data into the classes they should belong to. Optimal classification requires not only the correct separation but also the maximization of the separation distance (margin), in order to satisfy the structural risk minimization (SRM) notion. SVMs algorithm settles the problem from the simplest one, which is 2-dimension (binary) linear classification. However, for most complex data, it is hard to find the linear hyperplane in low-dimension therefore they are mapped into higher dimension, which will increase the computational complexity dramatically. Therefore, a Kernel function is introduced, in order to reduce the computation problem. Hence, proper choice of kernel function helps getting the classification function in higher dimension classifier much easier.

Figure 2.4: A binary classification problem (+ vs. -) in two dimensions. [21]

2.4.2 Software introduction

- **SVMlight [22]**

SVMlight is an implementation of Support Vector Machines that solves the problems of pattern recognition classification, regression and ranking function. The system introduces two main functions for classification, training and classifying. The software was implemented in C. However, there are interfaces in many other programming languages, such as PERL, Matlab, Python, DLL, Java, etc. SVMlight consists of a learning module (SVMs_learn) and a classification module (SVMs_classify) [23]. The classification module can be used to apply the learned model to new examples, which is adopted in our project for the Step 3 (mentioned in Chapter 3.2) here.

- **Learning from Positive and Unlabeled examples (LPU) [23]**

LPU (which stands for learning from positive and unlabeled examples) is a text learning and classification systems that learns from a set of positive documents and a set of unlabeled documents (without labeled negative documents) [23].
With the characteristics of LPU, we can handle the problem we met at the beginning of our task, which is that there is no negative example among the documents we want to align. The key technique of LPU that is different from other classification methods is that the unlabeled sets are treated as negative data. By iteratively applying classification algorithm, SVMa or EM, more reliable negative data can be identified from unlabeled set.
Chapter 3

Analysis, Design and Implementation

In this Chapter, we will analyze the feasibility of each software and algorithm, as well as present each step of the system design and implementation, including the comparison study of algorithms, the alternative method and the manual for users.

3.1 Analysis of Algorithm and software

3.1.1 Basic Algorithm

The instance-based strategy requires plenty of instances as its name suggested, since the accuracy of the results will be influenced by the quantity of primitive data, according to the statistic theory. Hence, in this thesis project we need to obtain instances (documents) for the concepts in GO ontologies, Signal-Ontology, Medical Subject Headings (MeSH) and the Anatomical Dictionary for Adult Mouse (MA). With these documents, SVMS algorithm is applied in Section 3.2 for classifier generating. To realize SVMS algorithm in this project, we are aiming at generating one classifier for each single concept, and classifying other concepts in other ontologies as mentioned in Section 3.3. The final similarity calculation is described in Section 3.4.

![Figure 3.1: A general alignment algorithm workflow](image)

3.1.2 Document-based ontology alignment with NB Algorithm

Since the new algorithm is proposed based on the same strategies as in previous work,
we present here a general introduction of the algorithm [1] where Naive Bayes algorithm is implemented and evaluated with some biomedical ontologies.

In [1], corpora are generated by querying PubMed with the concept names in the ontologies and retrieving the abstracts of the documents that contain the query terms. PubMed is a service of the National Library of Medicine that includes 18 million citations from MEDLINE and other biomedical journals. Version October 23, 2005 of PubMed was queried and different maximum numbers of abstract were retrieved for each concept for each time, though, there may not be as many abstracts as the total number required.

In Step 2, one classifier is generated corresponding to one ontology based on Naive Bayes algorithm. The classifiers return for a given document d the concept C in the ontology for which the posterior probability P(C|d) results in the highest value.

In Step 3, all the documents of Ontology A are classified by the classifier of Ontology B and vice versa.

In Step 4, similarity value between concepts from different ontologies is computed in the end. Similarity between concept C1 from Ontology A and C2 from Ontology B is defined as follow:

\[
\text{sim}(C_1, C_2) = \frac{n_{NB_C2}(C_1, C_2) + n_{NB_C1}(C_2, C_1)}{n_D(C_1) + n_D(C_2)}
\]

where \(n_D(C)\) is the number of abstracts originally associated with C, and \(n_{NB_Cx}(C_p, C_q)\) is the number of abstracts associated with \(C_p\) that are also related to \(C_q\) as found by classifier NBCx related to ontology x.

### 3.1.3 Document-based ontology alignment with SVMs Algorithm

#### 3.1.3.1 Why should SVMs work well for ontology alignment?

Talking about whether it is proper to implement SVMs in ontology alignment, we need to specify the core technique in ontology alignment first. Since we have the example of ontology alignment with instance-based strategy using Naive Bayes algorithm [1], we realized that the key step for document-based strategy is classifier generation and classification. Moreover, the instances we used are textual abstracts from PubMed. Therefore, the issue we need to discuss becomes whether it is proper to implement SVMs in document classification.

Obviously, there are many articles and systems had proved that Support vector machines algorithm performs well for text classification [21]. Since the properties of
text meet the principles of SVMs very well. Text or document includes such properties:

- High dimensional input space:
- Few irrelevant features:
- Document vectors are sparse:
- Most text classification problems are linearly separable:

### 3.1.3.2 Comparison of Software for SVMs

Nowadays, there is a number of mature software implementing SVMs classification, regression and other functions available on the internet. Most of them focus on different requirements and solutions. However, SVMs algorithm is focused and employed as the kernel algorithm in most of the software. They are constructed by various programming languages like C, C++, Matlab, Java and so forth. And each has its strong point as listed below:

<table>
<thead>
<tr>
<th>Name</th>
<th>Author</th>
<th>Language</th>
<th>Execution Environment</th>
<th>UI</th>
<th>Classification</th>
<th>Regression</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BSVMS</strong></td>
<td>Chih-Wei Hsu and Chih-Jen Lin</td>
<td>C++</td>
<td>Windows</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>LIBSVMS</strong></td>
<td>Chih-Chung Chang, Chih-Jen Lin</td>
<td>C++, Java, Python, R, MATLAB, Perl, Ruby</td>
<td>Win/*nix</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>mySVMS</strong></td>
<td>Stefan Ruping</td>
<td>C++</td>
<td>Win/*nix</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Matlab SVMS Toolbox</strong></td>
<td>S. R. Gunn</td>
<td>Matlab</td>
<td>N/A</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Statistical Pattern Recognition Toolbox for MATLAB</strong></td>
<td>Vojtech Franc and Vaclav Hlavac</td>
<td>Matlab/C</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td><strong>SVM^light</strong></td>
<td>Thorsten Joachims</td>
<td>C</td>
<td>Win/*nix</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>LPU</strong></td>
<td>Bing Liu, Xiaoli Li</td>
<td>Win</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
Table 3.1 Comparison of software for SVMs

Table 3.1 presents some information of the software developed from SVMs, also some comparisons for different purpose. Having studied of the software above, we select SVM\textsuperscript{light} and LPU as the implementation for our target, document-based classification for ontology alignment.

3.2 Alignment Algorithm Design

In this section, we focus more about the SVMs algorithm in text classification with instance-based strategy for ontologies alignment.

3.2.1 Basic Algorithm

- Generating corpora

The ontologies we want to align are represented as a group of corpora which is generated in the previous work [1].

- Generating classifiers

Unlike the Naive Bayes algorithm which generates a classifier per ontology, SVMs algorithm produces classifiers for each concept in each ontology. We utilize two software: LPU [23] and SVM\textsuperscript{light} [22] for classifier generation and classification separately. SVM\textsuperscript{light} performs well in many areas such as classification, regression, pattern reorganization and ranking functions relying on a set of existed sample example which includes the positive and negative features. However, the abstracts we have are retrieved from some standard ontology which excluded unlabeled examples. In that case, LPU performs better in classifying unlabeled concepts, while SVM\textsuperscript{light} focuses more on classification with the classifiers generated by LPU.

- Classification

An SVMs classifier is generated for each concept, which means there are as many classifiers as single concepts for each ontology. The classification is performed alternately between ontologies. For each abstract in one ontology, every classifier from the other ontology is applied and vice versa. Implementations of the process above are executed by the system SVM\textsuperscript{light}.

- Generating similarity values

The similarity between concepts C1 from Ontology A and C2 from Ontology B, is
defined as:

\[ n_{\text{SVMC}-C_2}(C_1, C_2) + n_{\text{SVMC}-C_1}(C_2, C_1) \]

\[ - \frac{n_D(C_1) + n_D(C_2)}{2} \]

Where \( n_{DI} \) is the number of abstracts retrieved from PubMed associated with \( C \), and \( n_{\text{SVMC}-Cx}(C_y, C_x) \) is the number of abstracts associated with \( C_y \) and also \( C_x \). SVMC-Cx is the classifier of Concept Cx.

### 3.2.2 Alternative: plural and single classification

With our SVMs-based approach, documents can be classified to several concepts in the other ontology. We call this plural classification. With the NB-based method documents can only be classified to one concept. In order to evaluate the performance between Naive Bayes and Support vector machines, we propose an alternative classifier generation for SVMs based on the way NB works. We assign a document only to the concept for which its SVMs classifier generated the highest positive value for that document. In the case that more than one has the same value of highest one, one associate concept is chosen randomly. We call this single classification.

### 3.3 Implementation

Based on the foundation of the previous work [1], we use the document from four ontologies GO, Signal-Ontology, Medical Subject Headings (MeSH) and the Anatomical Dictionary for Adult Mouse (MA). Following the last edition of the system, we keep writing the program in Java, besides, MySQL is introduced here to manage huge amount of data produced in the test.

Nowadays, lots of present resources and tools can be used to make software development easier. In this case, Eclipse, MySQL and MySQL Tray Monitor are used for application development. The running environment should be Windows PC with Java running and compiling environment, JRE and JDK.

#### 3.3.1 Environment Set Up

- Downloading Eclipse from [Eclipse downloads](#)
- Import existing source code into the new project
- Download and install the proper release from [MySql database server](#)
- Download [MySql Connection/J](#) for connecting MySql and Eclipse
- Extract the file to `\jre\lib\ext` path then add the location to the Java classpath in the environment path.
- Downloading SVM\textsuperscript{light} windows version from `SVM\textsuperscript{light}` and extract it to the folder named EXE for later use.
- Downloading LPU from `LPU`, extract the zip file and copy the `.exe` file into folder EXE for later use. So far, there are three executable files: `lpu.exe`, `SVM_classify`, and `SVM_learn` in the EXE folder.
- Set the path for all the executable files in Environment Path: Add the full path of EXE folder which contains all the executable files to the environment path.

### 3.3.2. Data Formulation

- **LPU file format requirement**

  Besides the three executable files (the files in the EXE Folder), there are three data files needed for running an LPU classification, which are [23]:
  - `filestem.pos`: It contains all the positive training data (or documents).
  - `filestem.unlabel`: It contains all the unlabeled data.
  - `filestem.test`: It contains all the test data. Positive documents should have target +1 and negative documents should have target -1.

  Each of the following lines represents one training example and is of the following format [22]:

  ```
  <line> . =. <target> <feature>:<value> <feature>:<value> ... <feature>:<value>
  <target> . =. +1 | -1 | 0 | <float>
  <feature> . =. <integer> | "qid"
  <value> . =. <float>
  ```

- **SVM\textsuperscript{light} file format requirement**

  There are two modules in SVM\textsuperscript{light}. one is learning which has been done in LPU classification step. The other one is classification which will apply the learning model generated from LPU to new test examples.

  In classification mode, the target value denotes the class of the example. +1 as the target value marks a positive example, -1 a negative example, value zero indicating unknown respectively. The test examples are given in the same format as the training examples (possibly with 0 as class label). [22]
3.3.3 Data preparation and format transformation

In order to implement SVMs algorithm in our project and test it, firstly, we need to prepare the corpus from 3.2.1 to the required format. Following the steps below, for each concept, a classifier is generated.

**Step 1.** With method mkdirs(String sig, String Name, String Parent) in Class GetName, generate folders for the transformed data, LPU test data, test data for SVM\(^{light}\) and classification results.

<table>
<thead>
<tr>
<th>parameter</th>
<th>description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>sig</td>
<td>The name of ontology</td>
<td>G_, S_, MeSH etc.</td>
</tr>
<tr>
<td>name</td>
<td>the path of original corpora</td>
<td>“E:/java/data-2005-10/data-2005-10/behavior/100/GO”</td>
</tr>
<tr>
<td>parent</td>
<td>The absolute path where data to be saved after transformation</td>
<td>“E:/java/mydata/behavior/100/GO/GO_behavior”</td>
</tr>
</tbody>
</table>

**Table 3.2: specification of parameters**

**Step 2.** With method writeNewsVector(String fpath) in Class Document, after removal of stop words, interpunction and suffix, primitive corpora are transformed into feature vector format presented as feature: value, fpath is a combination of sig, the main directory and the name of subdirectories.

**Step 3.** With the method writeVector(String outfile) in class Document, word stems are created, where outfile is the output path.
Figure 3.2: Word stem with frequency of occurrence

Also, we can query all the results from DOS with the SQL query language as shown in Figure 3.3.

Figure 3.3: query results from terminal
Step 4. With Class DictionaryCreate, table of word stems are created, meanwhile words are loaded to the database. The name of the table and path of the word stem should be aware and changed for different concepts.

Step 5. In Class TableCreate, first we create table for all the documents by method Create(String filePath), where the filePath is the directories where the feature vectors are saved, then load the data into each table when the file has the same name as table by method loadfile(String filePath), last, execute select queries to generate one file in the format of feature:value, which is required by SVMlight and LPU by the method executeQuery(String filePath, String dic, String out), where the dic is the name of word stem and out is the path of query result.

Step 6. After achieving the required form of input corpora, we start preparing each input for LPU and SVMlight. In Class FileMake, we start from substep 1, generate positive file with method posFile(String in, String Path,String out); substep 2, generating single file which is the corpora of all the first lines of each sub-concept by method firstLines(String in, String path, String out); substep 3, generating the unlabeled file by method unlabelFile(String lines, String folders); substep 4, generating the .test file for LPU by target the first line of each positive file with “+1”, using the method lputest(String in); substep 5, generating the test data for SVMlight, by target the .unlabel corpora generated from the subconcepts from Ontology 2 and...
the word stem in Ontology 1 with “0”, using the method svmtest2(String in, String Path, String out).

3.3.4 Calling external command in Java.

In Class Executefile, we call command LPU and SVMlight by method lpumodel(String path, String modelpath), path is the directory where all the .pos, . unlabel and .test data for LPU are. Method SVMclassify1(String in, String out, String prediction) and svmclassify2(String in, String out, String prediction) are two classification method for ontology 1 and 2, in is the path for LPU data, out is the path where SVMs' light test data is and prediction is the path to put all the results.

3.3.5 Calculation

Finally, in Class SimCal, we calculate the similarity between two ontologies for each subconcept. With method SimC(String input1, String input 2, String folder1, String folder2, String out), input 1 and 2 are the classification results, folder 1 and 2 are the path where vector form of saved and the out is the place to output the final similarity, similarities are calculated and output to a text file. With that text file we can convert it to an Excel file for later evaluation.
Figure 3.4: Workflow of classifier generation and classification.
Chapter 4

Evaluation

In order to verify the performance of a system, evaluation is required. In our evaluation phase, we focus on several aspects which might influence the performance in the end.

4.1 Description of the evaluation

We evaluate the proposed algorithm SVMs regarding the quality of the suggestions they generate. As suggested in the previous work [1], the number of abstracts on the quality of the suggestion is invested as a part of the evaluation. Also, the comparison of the single and plural assignment is carried on. Besides, comparison evaluation between SVMs and NB (Naive Bayes) [1] is deployed.

4.2 Evaluation procedure

Test cases: We keep the same five test cases generated from [1]. For the first two cases we use a part of a GO ontology [6] together with a part of SigO [24]. The first case, B (behavior), contains 57 terms from GO and 10 terms from SigO. The second case, ID (immune defense), contains 73 terms from GO and 17 terms from SigO. The other cases are taken from the anatomy category of Medical Subject Headings (MeSH, [25]) and the Adult Mouse Anatomy (MA, available from OBO): nose (containing 15 terms from MeSH and 18 terms from MA), ear (containing 39 terms from MeSH and 77 terms from MA), and eye (containing 45 terms from MeSH and 112 terms from MA). Golden standards for these cases were developed by domain experts.

Corpus: We use the same corpus generated in [1].

Matchers: In the evaluation we use SVM-based matchers as described in Chapter 3 to classify the concepts based on sets of maximum 20 and maximum 100 documents per concept. These matchers are denoted as SVM-20-P, SVM-100-P, SVM-20-S, and
SVM-100-S where the number stands for the maximum number of documents per concept in the corpus, and P and S stand for Plural (a document can be classified to several concepts) and Single (a document can be classified to only one concept), respectively.

### 4.3 Evaluation results

The following tables state the different evaluation results:

<table>
<thead>
<tr>
<th>Behavior</th>
<th>Threshold</th>
<th>SVM-20-P</th>
<th>SVM-100-P</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0.4</td>
<td>249/4/165/90</td>
<td>387/4/258/125</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>175/4/108/63</td>
<td>306/4/203/99</td>
</tr>
<tr>
<td></td>
<td>0.6</td>
<td>92/4/54/34</td>
<td>225/4/148/73</td>
</tr>
<tr>
<td></td>
<td>0.7</td>
<td>52/3/25/24</td>
<td>130/3/79/48</td>
</tr>
<tr>
<td></td>
<td>0.8</td>
<td>20/2/10/8</td>
<td>36/0/22/14</td>
</tr>
<tr>
<td>Defense</td>
<td>0.4</td>
<td>526/7/482/37</td>
<td>672/8/592/72</td>
</tr>
<tr>
<td>8</td>
<td>0.5</td>
<td>344/6/314/24</td>
<td>490/8/433/28</td>
</tr>
<tr>
<td></td>
<td>0.6</td>
<td>215/6/195/14</td>
<td>336/8/300/28</td>
</tr>
<tr>
<td></td>
<td>0.7</td>
<td>141/5/126/10</td>
<td>222/6/191/25</td>
</tr>
<tr>
<td></td>
<td>0.8</td>
<td>75/5/63/7</td>
<td>108/5/93/10</td>
</tr>
<tr>
<td>Nose</td>
<td>0.4</td>
<td>117/7/97/13</td>
<td>155/7/124/24</td>
</tr>
<tr>
<td>7</td>
<td>0.5</td>
<td>83/7/66/10</td>
<td>120/7/91/22</td>
</tr>
<tr>
<td></td>
<td>0.6</td>
<td>61/6/50/5</td>
<td>85/7/60/18</td>
</tr>
<tr>
<td></td>
<td>0.7</td>
<td>47/6/39/2</td>
<td>58/6/45/7</td>
</tr>
<tr>
<td></td>
<td>0.8</td>
<td>29/6/23/0</td>
<td>34/6/27/1</td>
</tr>
<tr>
<td>Ear</td>
<td>0.4</td>
<td>937/21/828/88</td>
<td>1224/24/1056/144</td>
</tr>
<tr>
<td>27</td>
<td>0.5</td>
<td>700/21/607/72</td>
<td>957/23/822/112</td>
</tr>
<tr>
<td></td>
<td>0.6</td>
<td>487/21/411/55</td>
<td>696/22/590/84</td>
</tr>
<tr>
<td></td>
<td>0.7</td>
<td>320/21/262/37</td>
<td>478/22/392/64</td>
</tr>
<tr>
<td></td>
<td>0.8</td>
<td>174/20/134/20</td>
<td>278/21/223/34</td>
</tr>
<tr>
<td>Eye</td>
<td>0.4</td>
<td>1588/25/1493/70</td>
<td>2055/25/1926/104</td>
</tr>
<tr>
<td>27</td>
<td>0.5</td>
<td>1089/25/1009/55</td>
<td>1481/25/1366/90</td>
</tr>
<tr>
<td></td>
<td>0.6</td>
<td>667/25/601/41</td>
<td>957/25/860/72</td>
</tr>
<tr>
<td></td>
<td>0.7</td>
<td>441/23/386/32</td>
<td>612/24/539/49</td>
</tr>
<tr>
<td></td>
<td>0.8</td>
<td>271/23/225/23</td>
<td>344/23/290/31</td>
</tr>
</tbody>
</table>

**Table 4.1: SVM plural assignment**
<table>
<thead>
<tr>
<th>Concept</th>
<th>Threshold</th>
<th>SVM-20-S</th>
<th>SVM-100-S</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Behavior</strong></td>
<td>0.4</td>
<td>2/2/0/0</td>
<td>0/0/0/0</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>2/2/0/0</td>
<td>0/0/0/0</td>
</tr>
<tr>
<td></td>
<td>0.6</td>
<td>2/2/0/0</td>
<td>0/0/0/0</td>
</tr>
<tr>
<td></td>
<td>0.7</td>
<td>2/2/0/0</td>
<td>0/0/0/0</td>
</tr>
<tr>
<td></td>
<td>0.8</td>
<td>2/2/0/0</td>
<td>0/0/0/0</td>
</tr>
<tr>
<td><strong>Defense</strong></td>
<td>0.4</td>
<td>4/3/0/1</td>
<td>2/2/0/0</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>3/2/0/1</td>
<td>0/0/0/0</td>
</tr>
<tr>
<td></td>
<td>0.6</td>
<td>2/1/0/1</td>
<td>0/0/0/0</td>
</tr>
<tr>
<td></td>
<td>0.7</td>
<td>0/0/0/0</td>
<td>0/0/0/0</td>
</tr>
<tr>
<td></td>
<td>0.8</td>
<td>0/0/0/0</td>
<td>0/0/0/0</td>
</tr>
<tr>
<td><strong>Nose</strong></td>
<td>0.4</td>
<td>9/5/4/0</td>
<td>5/5/0/0</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>5/5/0/0</td>
<td>4/4/0/0</td>
</tr>
<tr>
<td></td>
<td>0.6</td>
<td>5/5/0/0</td>
<td>2/2/0/0</td>
</tr>
<tr>
<td></td>
<td>0.7</td>
<td>4/4/0/0</td>
<td>0/0/0/0</td>
</tr>
<tr>
<td></td>
<td>0.8</td>
<td>2/2/0/0</td>
<td>0/0/0/0</td>
</tr>
<tr>
<td><strong>Ear</strong></td>
<td>0.4</td>
<td>18/16/2/0</td>
<td>14/12/2/0</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>15/15/0/0</td>
<td>11/10/1/0</td>
</tr>
<tr>
<td></td>
<td>0.6</td>
<td>11/11/0/0</td>
<td>1/1/0/0</td>
</tr>
<tr>
<td></td>
<td>0.7</td>
<td>5/5/0/0</td>
<td>0/0/0/0</td>
</tr>
<tr>
<td></td>
<td>0.8</td>
<td>2/2/0/0</td>
<td>0/0/0/0</td>
</tr>
<tr>
<td><strong>Eye</strong></td>
<td>0.4</td>
<td>17/16/1/0</td>
<td>7/7/0/0</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>14/14/0/0</td>
<td>4/4/0/0</td>
</tr>
<tr>
<td></td>
<td>0.6</td>
<td>9/9/0/0</td>
<td>0/0/0/0</td>
</tr>
<tr>
<td></td>
<td>0.7</td>
<td>2/2/0/0</td>
<td>0/0/0/0</td>
</tr>
<tr>
<td></td>
<td>0.8</td>
<td>0/0/0/0</td>
<td>0/0/0/0</td>
</tr>
</tbody>
</table>

**Table 4.2: SVM single assignment**

In the tables above, the first columns in the tables represent the concepts and the number of expected alignments for each concepts generated by the domain experts. For instance, there are 27 expected alignment suggestions that domain experts suggested. The expected alignments are a minimal set of suggestions that matchers are expected to generate for a perfect recall. The suggestion set does not include the inferred suggestions which are counted neither as correct nor as wrong suggestions. Name an inferred example, in biomedical and medical area incus is seen as a kind of ear ossicle. In this case we know that auditory bone (MA) is equivalent to ear ossicle (MeSH), and incus is a kind of auditory bone in MA. Then a reasoning mechanism could derive that incus is a kind of ear ossicle. The second column in the tables
represents threshold values. The similarity value higher than the threshold is considered to be a suggestion. The cells in the other columns contain quadruplets a/b/c/d which represent the number of a) suggestions, b) correct suggestions, c) wrong suggestions and d) inferred suggestions, for a given case, matcher and threshold.

Figure 4.1 Suggestion generation of Behavior-100

We may take a look at the example case that during the process of results generation. In Figure 4.1 above, case behavior is presented with every subconcepts in both ontologies GO and SO. The similarity value calculated by SVM-classifier is listed with each subconcept. S/IF/WS are stands for the suggested, inferred and wrong suggestions respectively.

4.3.1 Results analysis

Comparison based on different corpus sizes

For SVM with plural assignment (table 4.1 – columns SVM-20-P and SVM-100-P) we observe that SVM-20-P finds at most as many expected alignments as SVM-100-P, sometimes fewer. However, SVM-100-P does not only find more expected alignments, it also gives more suggestions in total, more wrong suggestions and more inferred
suggestions.

For SVMS with single assignment (table 4.2 – columns SVM-20-S and SVM-100-S) we observe that SVM-100-S does not find suggestions for high thresholds and for case B also not even for threshold 0.4. It does have almost perfect precision. SVM-20-S gives some more suggestions than SVM-100-S. It gives a few wrong suggestions at threshold 0.4, but no wrong suggestions at higher thresholds.

Comparison of single and plural assignments

Both for the SVM-100 and the SVM-20 versions the recall for the plural assignment is higher to much higher than the recall for the single assignment. This comes, however, at a cost. The precision for the single assignment algorithms is much higher than for their plural assignment counterparts. We see a real trade-off here: find many expected alignments, but also get many wrong suggestions, or, find few expected alignments, but receive almost no wrong suggestions.

Comparison of NB and SVM

NB is a single assignment algorithm and we therefore compare it with the single assignment versions of SVM. All single assignment algorithms give relatively few suggestions but have high precision. However, NB gives always more suggestions than SVM for the same document corpus and the same threshold. NB also always gives suggestions, except for case ID and threshold 0.8, while this is not the case for the SVM algorithms. SVM-20 does not give suggestions for ID - 0.7 and higher, and eye - 0.8. In addition to these, SVM-100 does not give suggestions for ID - 0.5 and higher, nose - 0.7 and higher, ear - 0.7 and higher, eye - 0.6 and higher, and B with threshold 0.4 and higher. It is clear that the SVM algorithms with single assignment do not perform well with high thresholds.

In general, the NB algorithms have slightly better recall than the SVMs algorithms, while the SVMs algorithms have slightly higher precision than the NB algorithms. However, even if the recall for NB is better, it is not always the case that the alignments that were found by SVMs algorithms were also found by NB.

<table>
<thead>
<tr>
<th>Behavior</th>
<th>Threshold</th>
<th>NB-20</th>
<th>NB-100</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0.4</td>
<td>3/2/0/1</td>
<td>4/2/1/1</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>2/2/0/0</td>
<td>2/2/0/0</td>
</tr>
<tr>
<td></td>
<td>0.6</td>
<td>2/2/0/0</td>
<td>2/2/0/0</td>
</tr>
<tr>
<td></td>
<td>0.7</td>
<td>2/2/0/0</td>
<td>2/2/0/0</td>
</tr>
<tr>
<td></td>
<td>0.8</td>
<td>2/2/0/0</td>
<td>1/1/0/0</td>
</tr>
<tr>
<td>Defense</td>
<td>0.4</td>
<td>11/4/4/3</td>
<td>9/6/3/0</td>
</tr>
<tr>
<td>8</td>
<td>0.5</td>
<td>7/4/0/0</td>
<td>5/5/0/0</td>
</tr>
<tr>
<td></td>
<td>0.6</td>
<td>5/4/0/1</td>
<td>2/2/0/0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>Nose</td>
<td>0.7</td>
<td>2/2/0/0</td>
<td>1/1/0/0</td>
</tr>
<tr>
<td>7</td>
<td>0.8</td>
<td>0/0/0/0</td>
<td>0/0/0/0</td>
</tr>
<tr>
<td></td>
<td>0.4</td>
<td>7/5/2/0</td>
<td>6/5/1/0</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>6/5/1/0</td>
<td>6/5/1/0</td>
</tr>
<tr>
<td></td>
<td>0.6</td>
<td>5/5/0/0</td>
<td>5/5/0/0</td>
</tr>
<tr>
<td></td>
<td>0.7</td>
<td>5/5/0/0</td>
<td>5/5/0/0</td>
</tr>
<tr>
<td></td>
<td>0.8</td>
<td>4/4/0/0</td>
<td>3/3/0/0</td>
</tr>
<tr>
<td>Ear</td>
<td>0.4</td>
<td>20/16/4/0</td>
<td>18/16/2/0</td>
</tr>
<tr>
<td>27</td>
<td>0.5</td>
<td>18/16/20</td>
<td>15/14/1/0</td>
</tr>
<tr>
<td></td>
<td>0.6</td>
<td>14/14/0/0</td>
<td>12/11/1/0</td>
</tr>
<tr>
<td></td>
<td>0.7</td>
<td>11/11/0/0</td>
<td>11/10/1/0</td>
</tr>
<tr>
<td></td>
<td>0.8</td>
<td>5/5/0/0</td>
<td>3/3/0/0</td>
</tr>
<tr>
<td>Eye</td>
<td>0.4</td>
<td>33/19/13/1</td>
<td>25/18/7/0</td>
</tr>
<tr>
<td>27</td>
<td>0.5</td>
<td>25/17/3/0</td>
<td>18/17/1/0</td>
</tr>
<tr>
<td></td>
<td>0.6</td>
<td>16/16/0/0</td>
<td>14/14/0/0</td>
</tr>
<tr>
<td></td>
<td>0.7</td>
<td>12/12/0/0</td>
<td>10/10/0/0</td>
</tr>
<tr>
<td></td>
<td>0.8</td>
<td>5/5/0/0</td>
<td>3/3/0/0</td>
</tr>
</tbody>
</table>

Table 4.3: Naive Bayes from [1]

### 4.3.2 Summary

Generally speaking, the more abstracts taken into consideration, the more suggestion we can get, on the other side, more inaccurate and inferred suggestions are generated at the same time. It means we need a balance on choosing the method considering the trade-off between recall and precision. Compared to Naive Bayes, SVMs algorithm has slightly better precision but slightly lower recall. [26]
Chapter 5

Conclusion and Future work

Thesis conclusion and prospect of related future work is included in this chapter.

5.1 Conclusion

In this thesis work, we extended the research in [1] by presenting a new classification method based on SVMs algorithm. Support vector machines algorithm has been proved successfully in many academic areas, which means a lot in other undiscovered domains. SVMs algorithm performs well in text classification due to its properties: (i) high dimensional input space, (ii) few irrelevant features, (iii) document vectors are sparse and (iv) most text classification problems are linearly separable. While most of the data resources of biomedical ontology alignment are the documents that related to the concepts we want to align.

5.2 Future work

The future research of ontology alignment might include:

- **Feasibility of different algorithms or combination in classifier generating based on instance for ontology alignment.**
  Following the basic concept of ontologies alignment, which is similarity comparison, there is a lot of possibility to implement this idea with different algorithms for classifiers generating.

- **Experiment on other type of test set**
  The instance in this paper is literature, which is qualified with the requirement of classification by SVMs algorithm. However, the instance in the real word could not be just one as text. Besides, the range could involve hundreds of domains except biomedical field. There is big chance to extend other type of instance in other domains.
- **Domain knowledge and other auxiliary knowledge development to improve the robustness of matching**
  With the exploring of domain knowledge, it could be possible to find out the way how entities were produced and generated. Studying the thinking behavior of human-being may help analyze the way ontologies were created. Those would improve the identification compared to relying on the suggestions of individual experts.

- **Trying to align more than two ontologies at one time to improve the efficiency of the system.**
## Appendix A

Application codes description

<table>
<thead>
<tr>
<th>Module</th>
<th>Java Code(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word stem creating and database setting</td>
<td>DictionaryCreate.java</td>
</tr>
<tr>
<td>External command calling Module</td>
<td>Executefile.java</td>
</tr>
<tr>
<td>Database setting of subconcepts Module</td>
<td>TableCreate.java</td>
</tr>
<tr>
<td>Similarity calculation Module</td>
<td>SimCal.java</td>
</tr>
<tr>
<td>Input file creation Module</td>
<td>FileMake.java</td>
</tr>
<tr>
<td>Word stem creation Module</td>
<td>Porter.java</td>
</tr>
<tr>
<td>Data format transformation Module</td>
<td>FileMake.java</td>
</tr>
</tbody>
</table>

**Table A**: Java Codes Description
Appendix B

List of Figures

Figure 2.1 example of overlapping ontologies[5]...................................................9
Figure 2.2: A general alignment strategy [12] .........................................................10
Figure 2.3: a general view of ontologies alignment................................................13
Figure 2.4: A binary classification problem (+ vs. -) in two dimensions. [21]....14
Figure 3.1: A general alignment algorithm workflow...............................................16
Figure 3.2: Word stem with frequency of occurrence...........................................23
Figure 3.3: query results from terminal .................................................................23
Figure 3.3: All the words appeared without repeat...............................................24
Figure 3.4: Workflow of classifier generation and classification. .......................26
Figure 4.1 Suggestion generation of Behavior-100 ...........................................30
Appendix C

List of Tables

Table 3.1 Comparison of software for SVMs ........................................ 19
Table 3.2: specification of parameters ............................................... 22
Table 4.1: SVMS plural assignment .................................................. 28
Table 4.2: SVMS single assignment ................................................... 29
Table 4.3: Naive Bayes from [1] ......................................................... 32
Table A : Java Codes Description ....................................................... 35
Bibliography


På svenska

Detta dokument hålls tillgängligt på Internet – eller dess framtida ersättare – under en längre tid från publiceringsdatum under förutsättning att inga extra-ordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för ickekommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns det lösningar av teknisk och administrativ art.

Upphovsmannens ideella rätt innefattar rätt att bli nämnd som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.

För ytterligare information om Linköping University Electronic Press se förlagets hemsida http://www.ep.liu.se/

In English

The publishers will keep this document online on the Internet - or its possible replacement - for a considerable time from the date of publication barring exceptional circumstances.

The online availability of the document implies a permanent permission for anyone to read, to download, to print out single copies for your own use and to use it unchanged for any non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional on the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its WWW home page: http://www.ep.liu.se/

©2008, Wei Xu