Semantic Analysis of Free Text and its Application on Automatically Assigning ICD-9-CM Codes to Patient Records

Ping Chen   Araly Barrera

Department of Computer and Mathematical Sciences
University of Houston-Downtown
Houston, Texas, USA

Chris Rhodes

Department of Computer Science
University of Central Arkansas
Conway, Arkansas, USA

Abstract

Free text written in natural languages is lack of definite structures, and semantic analysis of such text is critically important to many Natural Language Processing applications. In this paper, based on dependency parsing, we present a semantic analytic technique to automatically assign clinical ICD-9-CM codes to complex medical patient records written in English. Our semantics analysis method includes dependency parsing of clinical records obtained from testing and training data sets and the calculation of semantic matching scores. Our ultimate goal is to develop an automated disease diagnosis tool through accurate clinical code assignments. We evaluated our technique with a real-world corpus utilized in the 2007 International Natural Language Processing Challenge administered by the Computational Medicine Center at the Cincinnati Children's Hospital, and obtained promising results.

Keywords: *Semantic analysis, Dependency parsing, Natural Language processing, Automatic Diagnosis*

1. Introduction

Modernizing and digitalizing health care information is critical to lower health care cost and improve health care quality. However, the obstacles spawned from processing large amounts of clinical free text (also known as clinical coded free text) are numerous, which can impose serious problems to medical facilities such as clinics and hospitals. Although Natural Language Processing techniques have been widely applied in health care industry [10], many issues still remain unsolved, and the major obstacle is terabytes of text that clinical facilities have to save, index, and analyze every day [3].

Among the huge amount of patient information, this paper specifically aims at patient records like the following simplified example (the actual record format will be discussed in Section 4.1):

786.2
5-year-old male with cough, normal slightly hypoventilatory chest x-ray, no pneumonia

This record includes two parts: ICD-9-CM code (786.2) and symptoms of a patient. ICD-9-CM (International Classification of Disease 9th Revision Clinical Modification) codes are disease classification representation codes that are stored in patient records after a medical doctor gives a diagnosis based on symptoms of a patient [8]. Currently these clinical codes are widely used in medical settings that work to reduce the amount of text for both storage and processing improvements and for justifications of medical procedures to be performed to a patient. ICD-9-CM code is typically a 3 to 5 digit number with a decimal point after the third digit and organized in hierarchal levels. The highest levels usually represent a range of general medical conditions as the lower levels would give more specific details on a condition. Here is an example of an ICD-9-CM code and its representing condition:

DISEASES OF THE RESPIRATORY SYSTEM*, ICD-9-CM code range: **[460-519]***

→***PNEUMONIA AND INFLUENZA****, ICD-9-CM code range: **[480-488]***

→***INFLUENZA*** *due to identified avian influenza virus, ICD-9-CM code:* ***[488.0]***

In this example, any disease due to the respiratory system would be classified under the range of codes 460-519, and as the level goes lower, the more specific the particular condition becomes, e. g.,

"Pneumonia and influenza" are represented with codes in [480-488], and "influenza due to an avian influenza virus" is represented and assigned with code 488.0.

In addition to medical facilities, insurance companies also use these ICD-9-CM codes to determine if customers' medical expenses are covered by their policies. For some insurance companies, certain number of digits must be present in the code in order for proper reimbursements to be made to the customer [7].

The particular sets of ICD-9-CM codes used in this project are gathered from radiology reports (records that contain diagnosis after medical-imaging tests) which are assigned based on chest x-ray and renal procedure (e. g., kidney tests) results. These records are composed of clinical-free, raw text that medical doctors or radiologists would record about conditions of patients. The goal of this project is to automatically generate the corresponding ICD-9-CM codes for these radiology reports. Such an automatic and efficient approach can dramatically reduce the time and efforts to index, process, and retrieve patient information. More importantly, our method can be applied to perform cross-validation with ICD-9-CM codes generated from physicians and medical personnel to weed out potential errors and improve diagnosis accuracy.

The main contribution of this paper is the deep-level semantic analysis approach that involves dependency parsing, parse tree matching, and semantic matching score calculation between training and testing corpus. Moreover, as an unsupervised method, our approach does not need any manual intervention, which overcomes the knowledge acquisition bottleneck suffered in many existing methods. The paper is organized as follows. Related work is discussed in Section 2. In Section 3, we will present our semantic analysis technique including data format and corpus, dependency parsing, parse tree matching and score calculation. In Section 4 we present the evaluation results based on a real-world challenge corpus, and we conclude in Section 5.

2. Related Work

The Computational Medicine Center at the University of Cincinnati provided training corpus and testing corpus for the 2007 International Medical Natural Language Processing Challenge [1], and totally 44 systems participated in the competition. It is important to note that primary purpose of this challenge is to seek techniques to convert free texts into actionable knowledge for the advancement of the research scientific field and for the simulations of physician decision-making process [4].

Currently there are two types of approaches to automate the assignment of ICD-9-CM code to clinical free text:

1) Keyword-based methods

Keyword-based methods try to match keywords in training corpus and testing corpus. While this simple approach can assign codes quickly, no deep semantic analysis is performed, which inevitably introduces errors. For example, both training text and testing text can contain "cough", but in training text it could be "A 5-year old has serious cough." In testing text, "The patient does not cough anymore." Apparently treating "cough" as a match in this case is a mistake. Additionally in many natural languages a word can have multiple meanings, so even an exact matching of words does not necessarily means the same thing, which has been studied under Word Sense Disambiguation in Computational Linguistics. Also the same meaning can be phrased in many different ways. All of these issues cannot be addressed by keyword-based approaches, and significantly hurt its performance.

2) Rule-based methods

Rule-based methods incorporate knowledge acquired from human experts into the code assignment process, and these knowledge is usually expressed with a set of proposition rules. This type of methods achieved the best results in 2007 International Medical Natural Language Processing Challenge [9], and all of the best three performing systems are rule-based as shown in Table 1. However, based on decades of research in Artificial Intelligence, rule-based methods suffer from the knowledge acquisition bottleneck, and especially when the rule base grows, maintenance of the rule base becomes increasingly difficult due to conflicts and overlapping among rules.

In this paper we take an automatic approach that acquires knowledge directly from training corpus without manual intervention, and then apply these knowledge in the code assignment process. The details will be explained in the next section.

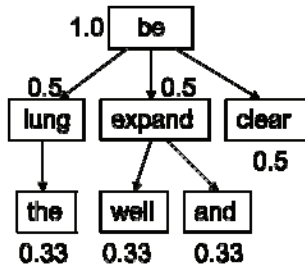## 3. Dependency Parsing-based Semantic Analysis

Figure 2 Parse tree of "The lungs are well expanded and clear." (the values beside each node are the weights of the nodes and will be explained in the next section.)
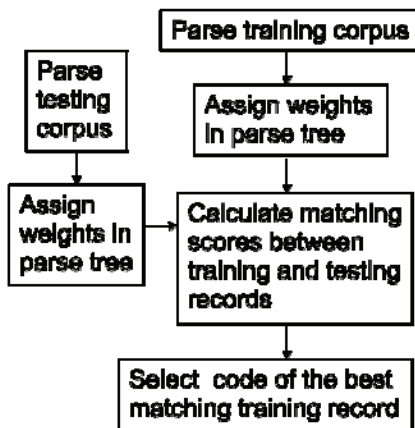

Figure 1 Our Semantic Analysis and Code Assignment System Architecture

Figure 1 shows the overall architecture for our semantic analysis and code assignment system. The system runs as follows. First, both training corpus and testing corpus are parsed with a dependency parser. Then nodes in the parse trees are assigned with different weights to show their respective contributions to the overall semantics of the whole sentence. Each parsed patient record from testing corpus is matched with every parsed training record, and a matching score is generated. Finally the ICD-9-CM code of the training patient record that generates the highest matching score is assigned to the testing patient record. Along this process, dependency parsing, weight assignment, and semantic matching score calculation deserve more discussion and will be explained in the following sections.

3.1 Dependency Parsing

Lexical dependency knowledge consists of dependency relations generated by dependency parsing of text. A dependency relation is an asymmetric binary relation between two words, one called head or governor, and the other called

dependent or modifier [6]. In dependency grammars a sentence is represented as a set of dependency relations, which normally form a tree that connects all the words in a sentence. For example, "blue sky" contains one dependency relation: "sky -> blue", where "sky" is the head, and "blue" is the dependent. Lexical dependency knowledge can be used in many lexicon-level Natural Language Processing applications. For example, parsing "Colorless green ideas sleep furiously" will generate the following dependency relations, "idea -> green", "idea -> colorless", "sleep -> idea", and "sleep -> furiously", and we can conclude that the sentence makes totally no sense since none of its dependency relations are semantically valid. Similarly dependency knowledge can also be used to catch spelling errors by checking semantic dependency relation violations. Another application is word prediction and automatic completion by identifying connected words. Potentially lexical dependency knowledge can improve the performance of many Natural Language Processing applications.

We adopt the dependency parser, Minipar, in our project. Minipar has been widely used in Computational Linguistics and Natural Language Processing. An evaluation with the SUSANNE corpus shows that Minipar achieves 89% precision with respect to dependency relations [5]. After parsing sentences are converted to parse trees and saved in files. For example, here is the description from one patient record:

"97644541: Five year old with cough. The lungs are well expanded and clear. There is no focal infiltrate or pleural effusion. The cardiac and mediastinal silhouette is normal. No bony abnormalities are seen."

After parsing, Minipar outputs all dependency relations extracted from a sentence. For example, here is the output after parsing *"The lungs are well expanded and clear."*

start nodes	(gov	end nodes)
1. *The*	*(gov*	*lung)*
2. *lung*	*(gov*	*be)*
3. *and*	*(gov*	*expanded)*
4. *well*	*(gov*	*expanded)*
5. *clear*	*(gov*	*be)*
6. *expand*	*(gov*	*be)*

Dependencies generated by Minipar consist of governing and governed word pairs, which are indicated with syntax: *"word₁ (gov word₂)"* where $word_1$ is considered a descriptive word to $word_2$ and

is therefore considered "governed" by $word_2$ in the context of sentence it is in. We considered all governed words as *start nodes* and all governing words as *end nodes*. To better illustrate the parsing result we also render the parse tree visually in Figure 2.

3.2 Assigning Weights to Nodes in Parse Trees

As for any form of text, the words in natural languages are all insignificant to computers without the interpretation of some kind of semantic representation to them. To serve as an analogy to this problem, a medical doctor gives his diagnosis to a patient based on what they know on a particular medical condition and many symptoms involved in people. The doctor bases his judgment on the whole knowledge he has on the condition and determines the best possible cause and solution to a patient's problem. In this case, the knowledge that we give our computer program is based on the words extracted from the clinical sentences and on the numerical weights we assign to them. After parse trees are generated for each sentence in every patient record, a weight is assigned to each node within the testing and training parse trees to distinguish the importance levels of these nodes. Based on the level of each node in the parse tree, its weight is given as:

$$weight = \frac{1}{level}$$

Larger weights will be given to the nodes in the higher level of parse trees. Usually the root of a parse tree is a verb, and the largest weight of "1" is assigned to the root, which has been validated by findings in psychology research. It is generally recognized in most linguistic theories and psycholinguistic models of sentence comprehension that the main determinant of sentence meaning is the root verb [2].

3.3 Semantic Score Calculation

In this step we will match the parse trees of the sentences from testing and training document and generate a score. The training documents are then ranked according to their scores, and the ICD-9-CM code of the training patient document that generates the highest matching score is assigned to the testing patient document.

Matching the testing document and training document consists of matching each testing sentence with each training sentence and generating a similarity score between them. So first let's focus on the score calculation process between two sentences.

The calculation process starts with an edge matching between two sentences. If any edges match then the edges are recursively linked together if possible. Finally scores are calculated for the largest possible structure and any remaining nodes should they exist. Here is the score calculating algorithm between one training sentence and one testing sentence:

1. For one sentence in testing document, load its parse tree;
2. For one sentence in training document, load its parse tree;
3. Search for the common nodes between the parse trees of testing and training sentences;
4. Search for the common edges between the parse trees;
5. Recursively search for the common large structures (including more than one edge) between the parse trees;
6. Assign score to all common nodes, edges, and structures, calculate the total score;

Here is the detailed step-by-step procedure:

Step 1: Load parse trees of one training sentence and one testing sentence, and each node from a testing sentence is compared to every node in a training sentence for matching.

Step 2: If $word_1$ from testing sentence matches with $word_1'$ from training sentence, a score value is calculated by multiplying the matched words' weights:

$$single\ node\ match\ score = (w1 * w1')$$

Where *w1* represents the weight of $word_1$ and *w1'* represents the weight of $word_1$'. Single word matching scores for all matched words in the sentence pair are then added into a single match score value, Score 1:

$$Score\ 1 = \sum (wn * wn')$$

where *Score 1* would hold the sum of all the single word matches between the testing sentence and the training sentence.

Step 3: Next we will calculate the edge matching score. If the testing parse tree and training parse tree have the same dependency edge $word_1$-> $word_2$, the following formula is used for calculating edge matching scores:

$$edge\ match\ score = (w1 + w2) \times (w1' + w2')$$

where *w1* represents the weight of $word_1$ and *w1'* represents the weight of $word_1$', *w2* represents the weight of $word_2$ and *w2'* represents the weight of $word_2$'. Edge matching scores for all

comparisons in the sentence pair are then added into a second score:

$$Score\ 2 = \sum((w1 + w2) \times (w1' + w2'))$$

where *Score 2* would hold the sum of all the edge matching scores between the testing sentence and the training sentence.

Step 4: This process is repeated for larger structures (3-node matching, 4-node matching, etc.). A final matching score between the two sentences would then be calculated as:

$$Final\ Score = \sum Score\ n$$

where *Final Score* represents the matching score between the testing sentence and the training sentence, which shows the similarity between these two sentences.

Scores are calculated between every sentence from a testing document and every sentence from a training document. Suppose there are *n* testing sentences and *m* training sentences, then *n×m* scores will be generated from exhaustively matching pairs of testing and training sentences. All of these *n×m* scores will be added together, and this is the similarity score between the training document and testing document. This process is repeated between a testing document and every training document. The testing document will be assigned the ICD-9-CM code of the highest scored training document, and details will be discussed in next section.

3.4   ICD-9-CM Code Assignment

After the calculation of matching scores from each testing document to every training document, and ICD-9-CM code is then assigned to the testing document. The highest final score obtained from the calculations would be traced back to the training document it is derived from and its majority ICD-9-CM code annotation will be assigned to the testing document. Here is an example to illustrate this process.

… …
Training 97698012
Score: 4.00

Training 97698138
Score: 0.00

Training 97698505
Score: 92.00

Training 97698653

Score: 6.00

… …

Here, when matching the testing document 97648672 with each training document, training document 97698505 generates the highest score of 92.0. This therefore means that training document number 97698505 is the best match to the testing document. We then look at the training document 97698505 which is shown below:

97698505
1   5
786.2
786.2
786.2
786.09
786.2
1
$5-year-old male with cough
1
%Normal slightly hypoventilatory chest x-ray, no pneumonia

The majority ICD-9-CM code annotation provided in the training document 97698505 would then be assigned to the testing document 97648672 as the representation of the diagnosis, e.g., testing document 97648672 is diagnosed with code 786.2.

4.   Experiment
4.1   Data Collection

The corpus used for our experiments is provided by the Computational Medicine Center's Natural Language Processing Challenge, which consists of a training corpus and a testing corpus containing samples of actual patient records from the Department of Radiology in the Cincinnati Children's Hospital [1]. The training data file contains a total of 978 patient records and the testing data file with a total of 976 records. Each patient record listed in these files has been carefully anonymized and disambiguated in order to comply with U.S. health standards and to keep patient information confidential. For example, all male names have been replaced with "John", female names with "Jane", and all surnames with "Johnson." These files are XML formatted and contain ICD-9-CM codes that have already been diagnosed and assigned to clinical-free text by three different annotators: COMPANY1, COMPANY 2, and COMPANY 3. An ICD-9-CM code assignment is based on two aspects:

1) <u>A patient's clinical history</u> – these are sentences are written by a medical doctor before a patient is given a radiology procedure. These

sentences typically include a patient's symptoms along with their age and gender.

2) <u>Symptoms of a patient</u> – these are symptoms reported by a radiologist after a radiology procedure is performed and typically include any observations made from the test. For example, physical observations made from an x-ray would be considered an impression.

Since some instances involve annotators' assignments of more than one distinct code to a sample record, a majority annotation is provided as a "gold-standard" and used in this project as the principal assignment code. Here is a sample patient record from the corpus:

<doc id="97636670"
type="RADIOLOGY_REPORT">
    <codes>
    <code origin="CMC_MAJORITY" type="ICD-9- CM">786.2</code>
    <code origin="COMPANY3" type="ICD-9-CM">786.2</code>
    <code origin="COMPANY1" type="ICD-9-CM">204.0</code>
    <code origin="COMPANY2" type="ICD-9-CM">786.2</code>
    </codes>
    <texts>
    <text origin="CCHMC_RADIOLOGY" type="CLINICAL_HISTORY">Eleven year old with ALL, bone marrow transplant on Jan. 2, now with three day history of cough.</text>
    <text origin="CCHMC_RADIOLOGY" type="IMPRESSION">1. No focal pneumonia. Likely chronic changes at the left lung base. 2. Mild anterior wedging of the thoracic vertebral bodies.</text>
    </texts>
</doc>

As shown in the sample document above, an original document number is provided to serve as a document ID in both training and testing corpus. Clinical-free text is considered to comprise all sentences under "CLINICAL_HISTORY" and "IMPRESSION." Majority code annotations for these sentences are specified in the line containing "CMC_MAJORITY" and the rest are the individual annotations given by the three annotators.

4.2 Cleaning and Extraction

Before starting the assignment of ICD-9-CM codes to the clinical-free text obtained from the Training and Testing Data, it is first necessary to split all the patient records from these files into individual document files and to extract all required information. Pertinent information from each patient record consists of a patient record ID, Majority ICD-9-CM annotation(s), all other annotations, clinical history sentences, and the impression sentences. Once extracted from the XML-formatted Training and Testing files, these pieces of information are then recorded into individual files, which are titled after the original document ID and saved into the corresponding directory. This phase of the project is labeled as the "Cleaning and Extraction" stage, and here are the specific steps we have performed:

Step 1: Extract document ID, Majority Code annotations, individual annotations, clinical history sentences, and impression sentences.

Step2: Save these occurrences, along with a majority annotation count, total annotation code count, and clinical-free sentences into separate files, titled with original document ID number. These reformatted files are formatted as follows: document ID, majority annotation count, total annotation count, ICD-9-CM code annotations, clinical history sentence count, '$' clinical history sentences with punctuation marks, impression sentence count, and '%' impression sentences with punctuation marks.

Step 3: Store the reformatted document files into corresponding original data folder: either stored in Training Data or Testing Data file directory. All records contain at least one clinical history and impression sentence but must have contained both in order to be listed in the original Testing and Training XML files. We found that all of these original files are well-formatted, so a total of 978 Training documents are generated from Training corpus and a total of 976 from Testing corpus.

4.3 Results and discussion

We followed the algorithm described in Section 3. There are two ways to evaluate the assignment quality provided by the 2007 Natural Language Processing Challenge organizers, micro-average and macro-average F1 scores. Macro-averaging should be chosen when a classification method need perform consistently across all classes regardless of their distributions. On the other hand, micro-averaging may be preferred if classes of different density need to be weighted differently in the overall system performance (please refer to [1] for detailed discussion). We calculated our system performance using macro-averaging with the following formula:

$$\text{Average Accuracy}$$
$$= \frac{\text{Sum of indivisual } F1 \text{ score from all files}}{\text{Total number of Testing files}}$$

The resulting average accuracy percentage derived is 60.0**%**, and 585 of all testing patient records are correctly diagnosed. Table 1 shows the top three competitor results from the Computational Medicine Center Natural Language Processing Challenge along with our system.

Rank	System	Macro-average F1
1	Szeged	0.77
2	University at Albany	0.73
3	University of Turku	0.70
4	Our system	0.60

Table 1   System performance

All of these three top-performing systems are rule-based. While rule-based systems can achieve high-level performance within a restrained scale, they often are "brittle" in practice due to the knowledge acquisition bottleneck. On the other hand, our method needs practically no manual maintenance and intervention. Even though there is a gap between our current result and the best systems, some additional modifications and enhancements would be able to optimize our approach. For example, in our current method words that contained digits or any kind of non-alphabetic symbols are considered garbage words and are therefore eliminated. But these words could be useful, such as ages of patients. We will continue working on these issues and report our findings in the future.

5. Conclusion

Huge amount of patient records in hospitals and other medical facilities call for highly efficient and accurate assignment of ICD-9-DM code. In this paper we take free medical text containing specified symptoms made in radiology procedures as the foundation and the primary source of our diagnosis knowledge, and developed a highly automatic code assignment system. Based on our experiment using data provided in the 2007 Computational Medicine Challenge, it shows that the current weight and score calculation method is not enough to achieve the high assignment accuracy required in real-world setting. Yet, since no manual intervention is required in our approach, we do not suffer from the knowledge acquisition bottleneck and brittle performance, either. Through manual inspections of inaccurate diagnosis results it is found that original clinical free sentences from strongest-related testing and training documents do in fact have reasonably similar clinical data. Additional observations confirmed that our approach is indeed effective in relating strong similar document pairs together but not as efficient with correct clinical code assignments.

Acknowledgments

References

[1] Computational Medical Center. The Computational Medicine Center's 2007 Medical Language Processing Challenge. University of Cincinnati. 2007. Available at: www.computationalmedicine.org/challenge.

[2] A. Healy, G. Miller. The verb as the main determinant of sentence meaning. Psychonomic Science, 20, 372. 1970.

[3] M. Hurtado, E. K. Swift, and J. M. Corrigan. Crossing the Quality Chasm: A New Health System for the 21st Century. Institute of Medicine, National Academy of Sciences. 2001.

[4] D. Lang. Natural Language Processing in the Health Care Industry. Cincinnati Children's Hospital Medical Center, 2007.

[5] D. Lin. Dependency-based evaluation of minipar. In Proceedings of the LREC Workshop on the Evaluation of Parsing Systems, pages 234-241, Granada, Spain. 1998.

[6] I. Melcuk. Dependency syntax: theory and practice. State University of New York Press. 1987

[7] M. Moisio. A Guide to Health Care Insurance Billing. Thomson Delmar Learning, Clifton Park. 2000.

[8] National Center for Health Statistics. Classification of Diseases and Functioning Disability. Available at: www.cdc.gov/nchs/icd9.htm

[9] J. Pestian, C. Brew, and P. Matykiewicz. A Shared Task Involving Multi-label Classification of Clinical Free Text. Technical Report. Department of Linguistics, Ohio State University. 2007.

[10] O. Uzuner, P. Szolovits, and I. Kohane. i2b2 workshop on natural language processing challenges for clinical records. Proceedings of the Fall Symposium of the American Medical Informatics Association. 2006.