



uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES
ET POSTDOCTORALES



FACULTY OF GRADUATE AND
POSTDOCTORAL STUDIES

Md. Aminul Islam

AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M.C.S. (Master of Computer Science)

GRADE / DEGREE

School of Information Technology and Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Applications of Corpus-based Semantic Similarity and Work Segmentation to Database Schema
Matching

TITRE DE LA THÈSE / TITLE OF THESIS

Diana Inkpen

DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

Iluju Kiringa

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

Nathalie Japkowicz

Jean-Pierre Corriveau

Gary W. Slater

Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

Applications of Corpus-based Semantic
Similarity and Word Segmentation to Database
Schema Matching

by

Md. Aminul Islam

A thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
in partial fulfillment of the requirements for the degree of
Master in Computer Science

Ottawa-Carleton Institute for Computer Science
School of Information Technology and Engineering
University of Ottawa

© Md. Aminul Islam, Ottawa, Canada, 2006



Library and
Archives Canada

Bibliothèque et
Archives Canada

Published Heritage
Branch

Direction du
Patrimoine de l'édition

395 Wellington Street
Ottawa ON K1A 0N4
Canada

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file *Votre référence*
ISBN: 978-0-494-18429-5
Our file *Notre référence*
ISBN: 978-0-494-18429-5

NOTICE:

The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

AVIS:

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protègent cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.


Canada

Abstract

In this thesis, we present a method for database schema matching, the problem of identifying elements of two given schemas that correspond to each other. Schema matching is useful in e-commerce exchanges, in data integration/warehousing, and in Semantic Web applications. We first present two corpus-based methods: one method is for determining the semantic similarity of two target words and the other is for automatic word segmentation. Then we present a name-based element-level database schema matching method that exploits both the semantic similarity and the word segmentation methods. Our word similarity method uses Pointwise Mutual Information (PMI) to sort lists of important neighbor words of the two target words and distinguish the words which are common in both lists and aggregate their PMI values (from the opposite list) to calculate the relative similarity score. Our word segmentation method uses corpus type frequency information to choose the type with maximum length and frequency from “desegmented” text. It also uses a modified forward-backward matching technique using maximum length frequency and entropy rate if any non-matching portions of the text exist. For the database schema matching method, we also use normalized and modified versions of the Longest Common Subsequence (LCS) string matching algorithm with weight factors to allow for a balanced combination. We validate our methods with experimental studies, the results of which suggest that these methods can be a useful addition to the set of existing methods.

Acknowledgements

I have been privileged to work closely with a person I greatly admire, my supervisor Dr. Diana Inkpen. I am indebted to her for all her support, her guidance, and most of all her friendship. She is an outstanding supervisor and mentor. Grateful thanks to her, for having given me opportunity, ethics, and a sound research methodology.

I am grateful to my co-supervisor Dr. Iluju Kiringa for having given me the opportunity to do research in the evolving field of databases and guided me to the way of innovation and novelty.

I also thank to all the members of the NLP-chat group for their helpful suggestions and stimulating meetings.

I want to thank Thomas K. Landauer, Department of Psychology, University of Colorado, for providing the 80 TOEFL questions. I am grateful to Peter D. Turney, National Research Council of Canada, who provided the 50 ESL questions and Jayant Madhavan, University of Washington, for providing the experimental data set of schema matching.

I would also like to thank the members of my committee: Dr. Nathalie Japkowicz and Dr. Jean-Pierre Corriveau.

Table of Contents

ABSTRACT	II
ACKNOWLEDGEMENTS	III
TABLE OF CONTENTS	IV
LIST OF TABLES	VI
LIST OF FIGURES	VII
CHAPTER 1	1
INTRODUCTION	
1.1 GOALS OF THIS THESIS	1
1.2 CONTRIBUTIONS	2
1.3 OUTLINE	3
CHAPTER 2	4
CORPUS-BASED SEMANTIC SIMILARITY OF WORDS	
2.1 SEMANTIC SIMILARITY OF WORDS	4
2.2 RELATED WORK	5
2.2.1 <i>Latent Semantic Analysis (LSA)</i>	7
2.2.2 <i>PMI-IR</i>	8
2.3 PROPOSED SECOND ORDER CO-OCCURRENCE PMI METHOD	10
2.4 A WALK-THROUGH EXAMPLE	12
2.4.1 <i>Choosing the Values of β and γ</i>	12
2.5 EXPERIMENTAL RESULTS.....	16
2.6 SUMMARY.....	21
CHAPTER 3	22
CORPUS-BASED WORD SEGMENTATION	
3.1 WORD SEGMENTATION	22
3.2 RELATED WORK	24
3.3 PROPOSED WORD SEGMENTATION METHOD	26
3.4 A WALK-THROUGH EXAMPLE	30
3.5 EVALUATION AND EXPERIMENTAL RESULTS	34
3.6 SUMMARY	39

CHAPTER 4	40
NAME-BASED SCHEMA MATCHING IN DATABASES	
4.1 SCHEMA MATCHING	40
4.2 CLASSIFICATION OF SCHEMA MATCHING APPROACHES	41
4.3 PROPOSED NAME-BASED ELEMENT-LEVEL HYBRID SCHEMA MATCHING METHOD	46
4.3.1 <i>Choosing the values of ζ, λ and ς</i>	52
4.4 WALK-THROUGH EXAMPLES	52
4.4.1 <i>Example 1</i>	52
4.4.2 <i>Example 2</i>	54
4.5 EVALUATION AND EXPERIMENTAL RESULTS	56
4.6 SUMMARY.....	65
CHAPTER 5	66
CONCLUSION AND FUTURE WORK	
5.1 CONCLUSION.....	66
5.2 FUTURE WORK	67
BIBLIOGRAPHY	69

List of Tables

2.1	Sample texts after cleaning.....	13
2.2	Types and frequencies for the example.....	14
2.3	Bigram frequencies and the set X and the set Y of words with their PMI values.....	15
3.1	Sorted n -grams and their frequencies.....	32
3.2	Word segmentation performance of twelve search engines.....	38
4.1	Characteristics of the evaluation domains and our results.....	61

List of Figures

2.1	Results on the 80 TOEFL questions.....	17
2.2	Number of missing words on the 80 TOEFL questions.....	18
2.3	Results on the 50 ESL questions.....	18
2.4	Number of missing words on the 50 ESL questions.....	19
2.5	Correlation of noun pairs.....	19
3.1	Matching residue.....	28
3.2	Matching leftmax.....	29
3.3	Matching rightmax.....	30
3.4	Test result on the Brown corpus.....	36
4.1	Classification of schema matching approaches.....	42
4.2	Maximal consecutive LCS starting at character 1.....	50
4.3	Maximal consecutive LCS starting at any character n	51
4.4	Similarity matching.....	51
4.5	A sample schema named " 401carfinder " from <i>auto</i> domain.....	58
4.6	A sample schema named " AutoWeb " from <i>auto</i> domain.....	59
4.7	Manual mapping between " 401carfinder " and " AutoWeb ".....	60
4.8	Precision vs. similarity threshold curves of the two web domains for eleven different similarity threshold.....	62
4.9	Recall vs. similarity threshold curves of the two web domains for eleven different similarity threshold.....	63
4.10	P-R curves of the two web domains for eleven different similarity threshold.....	63
4.11	F-measure vs. similarity threshold curves of the two web domains for eleven different similarity threshold.....	64

Chapter 1

Introduction

1.1 Goals of This Thesis

Discovering semantically corresponding attributes in different database schemas is known as schema matching. Traditionally, the problem of matching schemas has essentially relied on finding pairwise-attribute correspondences. Schema matching is a core problem in e-commerce exchanges, in data integration/warehousing, and in Semantic Web applications. Also, schema matching is fundamental for enabling query mediation and data exchange across information sources (Seligman et. al., 2002). Schema matching problems arise both in “classical” scenarios such as company mergers, and in “new” scenarios such as the integration of diverse sets of queryable information sources over the Web.

We present a schema matching method that uses a single property (i.e., element name) for matching and achieves a comparable F-measure score compared to methods that use multiple properties (e.g., element name, text description, data instance, context description). If we use a single property instead of multiple properties, it can speed up the matching process; this is important when schema matching is used in peer-to-peer (P2P) data management systems or in online query processing environments. If the properties that we use for schema matching contain element names or any types of text description then we need to focus on both string matching and semantic similarity of words because

sometimes only string matching or only semantic similarity of words provides good mapping results, and sometimes we need to use both in a balanced way. Names in schemas are often not segmented (words are connected together to form a name); therefore a good word segmentation method is required for better schema matching results. We propose two corpus-based methods: one for determining the semantic similarity of words and the other for word segmentation; then we formulate a name-based schema matching method that uses these two corpus-based methods.

We were motivated to propose corpus-based similarity and word segmentation methods for the following reasons:

First, we focused our attention on corpus-based measures because of their large type coverage. The types that are used in real-world database schema elements are often not found in dictionary.

Second, off-the-shelf corpus-based similarity measures are not as comparable as dictionary-based measures in performance which drew us to devise a corpus-based similarity measure that would be comparable to dictionary-based measures in performance.

Third, some existing corpus-based word segmentation method provides good precision score, but provides low recall score and as a result low F-measure score. So, we were inspired to propose a corpus-based word segmentation method that would provide good F-measure score.

1.2 Contributions

In this thesis, we make the following contributions.

- First, we present a corpus-based method for determining the semantic similarity of two target words. Our method uses Pointwise Mutual Information (PMI) to sort lists of important neighbor words of the two target words and distinguish the words which are common in both lists and aggregate their PMI values from the opposite list to calculate the relative similarity score. Evaluation results show that our method outperforms several competing corpus-based methods.

- Second, we present a new corpus-based method for automatic word segmentation. Our method uses corpus type frequency information to choose the type with maximum length and frequency from “desegmented”¹ text. It also uses a modified forward-backward matching technique using maximum length frequency and entropy rate if any non-matching portions of the text exist. Evaluation result shows that our method outperforms several competing corpus-based segmentation methods.
- Third, we present a name-based element-level schema matching method that exploits our proposed corpus-based word similarity and word segmentation method together with a substring matching algorithm.
- Finally, we point out some areas where these methods or modified versions of these methods can be exploited.

1.3 Outline

The remainder of the thesis is organized as follows. Chapter 2 introduces the idea of corpus-based semantic similarity of words, and describes in detail our proposed method, with experimental results. In Chapter 3, we discuss corpus-based word segmentation and related work. We also describe our proposed word segmentation method with examples and evaluation. Then Chapter 4 presents database schema matching: a brief overview of schema matching approaches and our proposed name-based element-level hybrid schema matching method. Finally, we conclude in Chapter 5 with a brief discussion of future work.

¹ words are connected together to form a name

Chapter 2

Corpus based Semantic Similarity of Words

In this chapter we present a new corpus-based method for calculating the semantic similarity of two target words. By corpus we mean a large collection of texts. Our method, called Second Order Co-occurrence PMI (SOC-PMI), uses Pointwise Mutual Information to sort lists of important neighbor words of the two target words. We will use this method for the task of schema matching in Chapter 4.

2.1 Semantic Similarity of Words

Semantic relatedness refers to the degree to which two concepts or words are related (or not) whereas semantic similarity is a special case or a subset of semantic relatedness. Humans are able to easily judge if a pair of words are related in some way. For example, most would agree that *apple* and *orange* are more related than are *apple* and *toothbrush*. Budanitsky and Hirst (2004) point out that semantic similarity is used when similar entities such as *apple* and *orange* or *table* and *furniture* are compared. These entities are close to each other in an *is-a* hierarchy. For example, *apple* and *orange* are hyponyms of *fruit* and *table* is a hyponym of *furniture*. However, even dissimilar entities may be semantically related, for example, *glass* and *water*, *tree* and *shade*, or *gym* and *weights*. In this case the two entities are intrinsically not similar, but are related by some relationship. Sometimes this relationship may be one of the classical relationships such as meronymy (is part of) as in *computer* – *keyboard* or a non-classical one as in *glass* – *water*, *tree* – *shade* and *gym* – *weights*. Thus two entities are semantically related if they

are semantically similar (close together in the *is-a* hierarchy) or share any other classical or non-classical relationships.

Measures of the semantic similarity of words have been used for a long time in applications in natural language processing and related areas, such as the automatic creation of thesauri (Grefenstette, 1993; Lin, 1998; Li and Abe, 1998), automatic indexing, text annotation and summarization (Lin and Hovy, 2003), text classification, word sense disambiguation (Lesk, 1986; Yarowsky, 1992; Li and Abe, 1998), information extraction and retrieval (Buckley et al., 1995; Vechtomova and Robertson, 2000; Xu and Croft, 2000), lexical selection, automatic correction of word errors in text and discovering word senses directly from text (Pantel and Lin, 2002).

A word similarity measure is also used for language modeling by grouping similar words into classes (Brown et al., 1992). In databases, word similarity can be used to solve semantic heterogeneity, a key problem in any data sharing system whether it is a federated database, a data integration system, a message passing system, a web service, or a peer-to-peer data management system (Madhavan et al., 2005).

2.2 Related Work

Many different measures of semantic similarity between word pairs have been proposed, some using statistical or distributional techniques (Grefenstette, 1992; Lin, 1998), some using lexical databases (thesaurus), and some hybrid approaches, combining distributional and lexical techniques. PMI-IR (Turney, 2001) uses Pointwise Mutual Information (PMI) and Information Retrieval (IR) to measure the similarity of pairs of words. PMI-IR is a statistical approach that uses a huge data source: the web and the PMI of two words are approximated by the number of web documents where they co-occur. Another well-known statistical approach to measuring semantic similarity is Latent Semantic Analysis (LSA) (Landauer and Dumais, 1997). We will briefly discuss these two approaches in next sub sections.

Individual words in a given text corpus have more or less differing contexts around them. The context of a word is composed of words co-occurring with it within a certain window around it. Distributional measures use statistics acquired from a large text corpora to determine how similar the contexts of two words are. These measures are also

used as approximations to measures of semantic similarity of words, because words found in similar contexts tend to be semantically similar. Such measures have traditionally been referred to as measures of distributional similarity. If two words have many co-occurring words, then similar things are being said about both of them and therefore they are likely to be semantically similar. Conversely, if two words are semantically similar then they are likely to be used in a similar fashion in text and thus end up with many common co-occurrences. For example, the semantically similar *car* and *vehicle* are expected to have a number of common co-occurring words such as *parking, garage, model, industry, accident, traffic*, and so on, in a large enough text corpus.

Various distributional similarity measures were discussed in (Weeds et. al., 2004) where co-occurrence types of a target word are the contexts in which it occurs and these have associated frequencies which may be used to form probability estimates. Lesk (1969) was one of the first to apply the cosine measure, which computes the cosine of the angle between two vectors, to word similarity. The Jensen-Shannon (JS) divergence measure (Rao, 1983; Dagan et. al., 1999) and the skew divergence measure (Lee, 1999) are based on the Kullback-Leibler (KL) divergence measure. Jaccard's coefficient (Salton and McGill, 1983) calculates the proportion of features belonging to either word that are shared by both words. In the simplest case, the features of a word are defined as the contexts in which it has been seen to occur. Pointwise Mutual Information (PMI) was first used to measure word similarity by (Church and Hanks 1990) where positive values indicate that words occur together more than would be expected under an independence assumption and negative values indicate that one word tends to appear only when the other does not. Jaccard-MI is a variant (Lin, 1998) in which the features of a word are those contexts for which the pointwise mutual information between the word and the context is positive. Average Mutual Information corresponds to the expected value of two random variables using the same equation as PMI and was used as a word similarity measure by (Rosenfeld, 1996; Dagan et. al., 1999). Cosine of pointwise mutual information was used by (Pantel and Lin, 2002) to uncover word senses from text. L_1 norm method was proposed as an alternative word similarity measure in language modeling to overcome zero-frequency problems for bigrams (Dagan et. al., 1999). A

likelihood ratio was used by (Dunning, 1993) to test word similarity under the assumption that the words in text have a binomial distribution.

There are several dictionary-based approaches to measuring the similarity of words. Most of the dictionary-based approaches use WordNet (Miller, 1995), a broad coverage lexical network of English words. Some use Roget's Thesaurus. Budanitsky and Hirst (2004) presented a detail overview of several WordNet based measures. We briefly discuss Lin's (1998a) approach, one of the hybrid measures using both the WordNet and corpus in next subsection. Jarmasz and Szpakowicz (2003) implemented a similarity measure using Roget's Thesaurus.

2.2.1 Latent Semantic Analysis (LSA)

LSA (Landauer and Dumais, 1997), a high-dimensional linear association model, analyzes a large corpus of natural text and generate a representation that captures the similarity of words and text passages. The underlying idea is that the aggregation of all the word contexts in which a given word does and does not appear provides a set of mutual constraints that largely determines the similarity of meaning of words and sets of words to each other (Landauer et al., 1998). The model tries to answer how people acquire as much knowledge as they do on the basis of as little information as they get. It uses the Singular Value Decomposition (SVD) to find the semantic representations of words by analyzing the statistical relationships among words in a large corpus of text. The corpus is broken up into chunks of texts approximately the size of a text or paragraph or small document. Analyzing each text or paragraph, the number of occurrences of each word is set in a matrix with a column for each word and a row for each paragraph. Then each cell of the matrix (a word by context matrix, X), is transformed from the raw frequency count into the log of the count. After that each cell is divided by the entropy of the column, given by $-\sum p \log p$, where the summation is over all the paragraphs the word appeared.

The next step is to apply SVD to X , to decompose X into a product of three matrices

$$X = WSP'$$

where, W and P are in column orthonormal form (i.e., columns are orthogonal) and S is the diagonal matrix of non-zero entries (singular values). To reduce dimensions, the rows

of W and P corresponding to the highest entries of S are kept. In other words, the new lower-dimensional matrices W_L , P_L and S_L are the matrices produced by removing the columns and rows with smallest singular values from W , P and S . This new matrix

$$X_L = W_L S_L P_L'$$

is a compressed matrix which represents all the words and text samples in a lower dimensional space. Then the similarity of two words, using LSA, is measured by the cosine of the angle between their corresponding row vectors.

2.2.2 PMI-IR

PMI-IR (Turney, 2001), a simple unsupervised learning algorithm for recognizing synonyms, uses Pointwise Mutual Information as follows:

$$\text{score}(\textit{choice}_i) = p(\textit{problem} \ \& \ \textit{choice}_i) / p(\textit{choice}_i)$$

Here, *problem* represents the problem word and $\{\textit{choice}_1, \textit{choice}_2, \dots, \textit{choice}_n\}$ represent the alternatives. $p(\textit{problem} \ \& \ \textit{choice}_i)$ is the probability that *problem* and *choice_i* co-occur. In other words, each choice is simply scored by the conditional probability of the problem word, given the choice word, $p(\textit{problem} \ | \ \textit{choice}_i)$. If *problem* and *choice_i* are statistically independent, then the probability that they co-occur is given by the product $p(\textit{problem}) \cdot p(\textit{choice}_i)$. If they are not independent, and they have a tendency to co-occur, then $p(\textit{problem} \ \& \ \textit{choice}_i)$ will be greater than $p(\textit{problem}) \cdot p(\textit{choice}_i)$.

PMI-IR used AltaVista Advanced Search query syntax to calculate the probabilities. In the simplest case, two words co-occur when they appear in the same document:

$$\text{score}_1(\textit{choice}_i) = \frac{\text{hits}(\textit{problem} \ \text{AND} \ \textit{choice}_i)}{\text{hits}(\textit{choice}_i)}$$

Here, $\text{hits}(x)$ be the number of hits (the number of documents retrieved) when the query x is given to AltaVista. AltaVista provides how many documents contain both *problem* and *choice_i*, and then how many documents contain *choice_i* alone. The ratio of these two numbers is the score for *choice_i*. There are three other versions of this scoring equation based on the closeness of the pairs in documents, considering antonyms, and taking context into account.

2.2.3 Lin's Measure

Lin (1998a) noticed that most of the similarity measures were tied to a particular application domain or resource and then he attempted to define a similarity measure that would be both universal and theoretically justified. He used the following three intuitions as a basis:

- 1) The similarity between A and B is related to their commonality. The more commonality they share, the more similar they are. The commonality between A and B is measure by

$$I(\text{common}(A, B))$$

where $\text{common}(A, B)$ is a proposition that states the commonalities between A and B ; $I(s)$ is the amount of information contained in a proposition s .

- 2) The similarity between A and B is related to the differences between them. The more differences they have, the less similar they are. The difference between A and B is measure by

$$I(\text{description}(A, B)) - I(\text{common}(A, B))$$

where $\text{description}(A, B)$ is a proposition that describes what A and B are.

- 3) The maximum similarity between A and B is reached when A and B are identical, no matter how much commonality they share.

Given these assumptions and definitions and the apparatus of information theory, Lin proved the following theorem:

Similarity Theorem: The similarity between A and B is measured by the ratio between the amount of information needed to state the commonality of A and B and the information needed to fully describe what A and B are:

$$\text{sim}(A, B) = \frac{\log P(\text{common}(A, B))}{\log P(\text{description}(A, B))}$$

Lin demonstrated how this similarity theorem could be applied in different domains using WordNet and corpus. For example, his measure of similarity between two concepts in a taxonomy is a corollary of this theorem:

$sim(A, B) = \frac{2 \log P(Iso(A, B))}{\log P(A) + \log P(B)}$, where probabilities $P(x)$ are determined

by $P(x) = \frac{\sum_{w \in W(x)} count(w)}{N}$, where $W(x)$ is the set of words (nouns) in the corpus whose

senses are subsumed by concept x , and N is the total number of word (noun) tokens in the corpus that are also present in WordNet.

2.3 Proposed Second Order Co-occurrence PMI Method

Let W_1 and W_2 be the two words for which we need to determine the semantic similarity and $C = \{c_1, c_2, \dots, c_m\}$ denotes a large corpus of text (after some preprocessing e.g., stop words elimination and lemmatization) containing m words (tokens). Also, let $T = \{t_1, t_2, \dots, t_n\}$ be the set of all unique words (types) which occur in the corpus C . Unlike the corpus C , which is an ordered list containing many occurrences of the same words, T is a set containing no repeated words. Throughout this section, we will use W to denote either W_1 or W_2 .

We set a parameter α , which determines how many words before and after the target word W , will be included in the context window. The window also contains the target word W itself, resulting in a window size of $2\alpha + 1$ words. The steps in determining the semantic similarity involve scanning the corpus and then extracting some functions related to frequency counts.

We define the *type frequency* function,

$$f^t(t_i) = |\{k: c_k = t_i\}|, \text{ where } i = 1, 2, \dots, n$$

which tells us how many times the type t_i appeared in the entire corpus. Let

$$f^b(t_i, W) = |\{k: t_k = W \text{ and } t_{k+j} = t_i\}|,$$

where, $i = 1, 2, \dots, n$ and $-\alpha \leq j \leq \alpha$, be the *bigram frequency* function. $f^b(t_i, W)$ tells us how many times word t_i appeared with word W in a window of size $2\alpha + 1$ words.

Then we define *pointwise mutual information* function for only those words having $f^b(t_i, W) > 0$,

$$f^{pmi}(t_i, W) = \log_2 \frac{f^b(t_i, W) \times m}{f^t(t_i) f^t(W)},$$

where, $f^t(t_i)f^t(W) > 0$

and m is total number of tokens in corpus C as mentioned earlier. Now, for word W_1 , we define a set of words, X , sorted in descending order by their PMI values with W_1 and taken the top-most β_1 words having $f^{pmi}(t_i, W_1) > 0$.

$$X = \{X_i\}, \text{ where } i = 1, 2, \dots, \beta_1$$

$$\text{and } f^{pmi}(t_1, W_1) \geq f^{pmi}(t_2, W_1) \geq \dots \geq f^{pmi}(t_{\beta_1-1}, W_1) \geq f^{pmi}(t_{\beta_1}, W_1)$$

Similarly, for word W_2 , we define a set of words, Y , sorted in descending order by their PMI values with W_2 and taken the top-most β_2 words having $f^{pmi}(t_i, W_2) > 0$.

$$Y = \{Y_i\}, \text{ where } i = 1, 2, \dots, \beta_2$$

$$\text{and } f^{pmi}(t_1, W_2) \geq f^{pmi}(t_2, W_2) \geq \dots \geq f^{pmi}(t_{\beta_2-1}, W_2) \geq f^{pmi}(t_{\beta_2}, W_2)$$

Note that we have not yet determined the value for β 's (either β_1 or β_2) which actually depend on the word W and the number of types in the corpus (this will be discussed in the next section).

Again, we define the β -PMI summation function. For word W_1 , the β -PMI summation function is:

$$f^\beta(W_1) = \sum_{i=1}^{\beta_1} \left(f^{pmi}(X_i, W_2) \right)^\gamma,$$

where, $f^{pmi}(X_i, W_2) > 0$ and $f^{pmi}(X_i, W_1) > 0$

which sums all the positive PMI values of words in the set Y also common to the words in the set X . In other words, this function actually aggregates the positive PMI values of all the semantically close words of W_2 which are also common in W_1 . Note that we call it semantically-close because all these words have high PMI values with W_2 and this doesn't ensure the closeness with respect to the distance within the window size.

Similarly, for word W_2 , the β -PMI summation function is:

$$f^\beta(W_2) = \sum_{i=1}^{\beta_2} \left(f^{pmi}(Y_i, W_1) \right)^\gamma,$$

where, $f^{pmi}(Y_i, W_1) > 0$ and $f^{pmi}(Y_i, W_2) > 0$

which sums all the positive PMI values of words in the set X also common to the words in the set Y . In other words, this function aggregates the positive PMI values of all the

semantically-close words of W_1 which are also common in W_2 . We have not discussed the criteria for choosing the exponential parameter γ (this will be discussed in the next subsection).

Finally, we define the *semantic PMI similarity* function between two words, W_1 and W_2 ,

$$Sim(W_1, W_2) = \frac{f^\beta(W_1)}{\beta_1} + \frac{f^\beta(W_2)}{\beta_2}$$

2.3.1 Choosing the Values of β and γ

A rule of thumb is used to choose the value of β . The value of β is related to how many times the word, W appears in the corpus, i.e., the frequency of W as well as the number of types in the corpus. We define β as

$$\beta_i = (\log_2(f^t(W_i)))^2 \frac{(\log_2(n))}{\delta}, \text{ where } i = 1, 2 \text{ and } \delta \text{ is a constant.}$$

For all of our experiments we used $\delta = 6.5$. The value of δ depends on the size of the corpus. The smaller the corpus we use, the smaller the value of δ we should choose. If we lower the value of β we lose some important / interesting words, and if we increase it we consider more words common to both W_1 and W_2 and this significantly degrades the result.

γ should have a value greater than 1. The higher we choose the value of γ , the greater emphasis on words having very high PMI values with W . For all our experiments, we chose $\gamma = 3$.

2.4 A Walk-Through Example

Suppose we want to determine the semantic similarity between words $W_1 = car$ and $W_2 = automobile$ and the following 12 sentences (Table 2.1) are our corpus of text after preprocessing (stop-words elimination and lemmatization). Actually we are using a large corpus (the BNC), but we use this very small number of texts in this example to explain the method and this is the reason why most of the frequencies in Table 2.2 and PMI values in Table 2.3 are almost the same. Here, we have tokens $m = 70$ and types $n = 43$

(the types and the corresponding frequencies are in Table 2.2). The bigram frequencies for word W_1 and W_2 in a window of 11 words ($\alpha = 5$) are in Table 2.3.

For this small corpus, we have chosen $\delta = 0.7$ and $\gamma = 3$.

Here, $\beta_1 = (\log(f^t(W_1)))^2 \frac{(\log_2(n))}{\delta} = 24.88$,

and, similarly, $\beta_2 = 24.88$.

Now we determine the set X of words sorted in descending order by their PMI values with W_1 and take the top most 23 (β_1) (it could have been maximum 24) words (see Table 2.3) having $f^{pmi}(t_i, W_1) > 0$. Similarly, the number of words we got in Y is 19 (β_2), though it could have been at most 24.

1	pursuit accident claim car driver exclude
2	soak motorist company car driver risky
3	company car driver tend travel farther
4	job engineer disappear fall mechanical engineer car industry worst affect
5	sign recession car industry
6	brightest engineer moment car industry
7	yugoslavia benefit direct investment automobile industry
8	Acreage expand emergence automobile industry
9	automobile industry among hardest hit recession
10	automobile industry largely male force
11	component supplier automobile industry expand
12	client industry manufacturer component automobile industry

Table 2.1: Sample texts after cleaning

t_i	$f^i(t_i)$	t_i	$f^i(t_i)$
disappear	1	worst	1
yugoslavia	1	soak	1
pursuit	1	fall	1
brightest	1	supplier	1
travel	1	company	2
benefit	1	recession	2
risky	1	farther	1
sign	1	car	6
male	1	investment	1
accident	1	industry	10
affect	1	force	1
mechanical	1	job	1
claim	1	client	1
among	1	Tend	1
moment	1	hardest	1
engineer	3	component	2
automobile	6	manufacturer	1
emergence	1	expand	2
direct	1	driver	3
hit	1	exclude	1
largely	1		

Table 2.2: Types and frequencies for the example

X_i (also t_i)	$f^b(t_i, W_1)$	$f^{pmi}(t_i, W_1)$	Y_i (also t_i)	$f^b(t_i, W_2)$	$f^{pmi}(t_i, W_2)$
motorist	1	3.544	emergence	1	3.544
disappear	1	3.544	direct	1	3.544
worst	1	3.544	acreage	1	3.544
pursuit	1	3.544	hit	1	3.544
soak	1	3.544	largely	1	3.544
travel	1	3.544	yugoslavia	1	3.544
brightest	1	3.544	supplier	1	3.544
fall	1	3.544	benefit	1	3.544
risky	1	3.544	male	1	3.544
company	2	3.544	investment	1	3.544
sign	1	3.544	among	1	3.544
farther	1	3.544	force	1	3.544
accident	1	3.544	client	1	3.544
affect	1	3.544	hardest	1	3.544
mechanical	1	3.544	component	2	3.544
tend	1	3.544	manufacturer	1	3.544
claim	1	3.544	expand	2	3.544
engineer	3	3.544	industry	7	3.029
moment	1	3.544	recession	1	2.544
driver	3	3.544			
exclude	1	3.544			
recession	1	2.544			
industry	3	1.807			

Table 2.3: Bigram frequencies and the set X and the set Y of words with their PMI values

Then we compute:

$$f^\beta(W_1) = (f^{pmi}(\text{“recession”}, W_2))^y + (f^{pmi}(\text{“industry”}, W_2))^y = (2.544)^3 + (3.029)^3 = 44.255$$

Similarly,

$$f^\beta(W_2) = (f^{pmi}(\text{“industry”}, W_1))^y + (f^{pmi}(\text{“recession”}, W_1))^y = (1.807)^3 + (2.544)^3 = 22.364$$

Therefore,

$$\begin{aligned} Sim(W_1, W_2) &= \frac{f^\beta(W_1)}{\beta_1} + \frac{f^\beta(W_2)}{\beta_2} \\ &= \frac{44.255}{23} + \frac{22.364}{19} \\ &= 3.101 \end{aligned}$$

2.5 Experimental Results

Our method was empirically evaluated on the task of solving 80 synonym TOEFL questions and 50 synonym ESL questions; and using Miller and Charles’ (1991) 30 noun pairs subset and Rubenstein and Goodenough’s (1965) 65 noun pairs.

In literature, though most of the word similarity measures were evaluated using Miller and Charles’ (1991) 30 noun pairs subset and Rubenstein and Goodenough’s (1965) 65 noun pairs, we also evaluated our method on the 80 synonym TOEFL questions and 50 synonym ESL questions to judge how well our method performs on a task based test.

We computed the SOC-PMI similarity values using the BNC² as a source of frequencies and contexts. The size of this corpus is approximately 100 million words, and it is a balanced corpus: it contains texts from various sources, general British English.

Landauer and Dumais (1997) employed word similarity measures to answer 80 synonym test questions from the Test of English as a Foreign Language (TOEFL) using Latent Semantic Analysis (LSA). Turney (2001) applied his Pointwise Mutual Information and Information Retrieval (PMI-IR) measure to answer 50 synonym test questions from a collection of English as a Second Language (ESL) tests and the same 80 TOEFL questions set that Landauer and Dumais (1997) used.

For the 80 TOEFL questions, the SOC-PMI method correctly answered 76.25% of the questions, as shown in Figure 2.1. This is an improvement over the results presented by

² <http://www.natcorp.ox.ac.uk/>

Landauer and Dumais (1997), using LSA, where 64.5% of the questions were answered correctly, and Turney (2001), using the PMI-IR algorithm, where the best result was 73.75%. A human average score on the same question set is 64.5% (Landauer and Dumais, 1997).

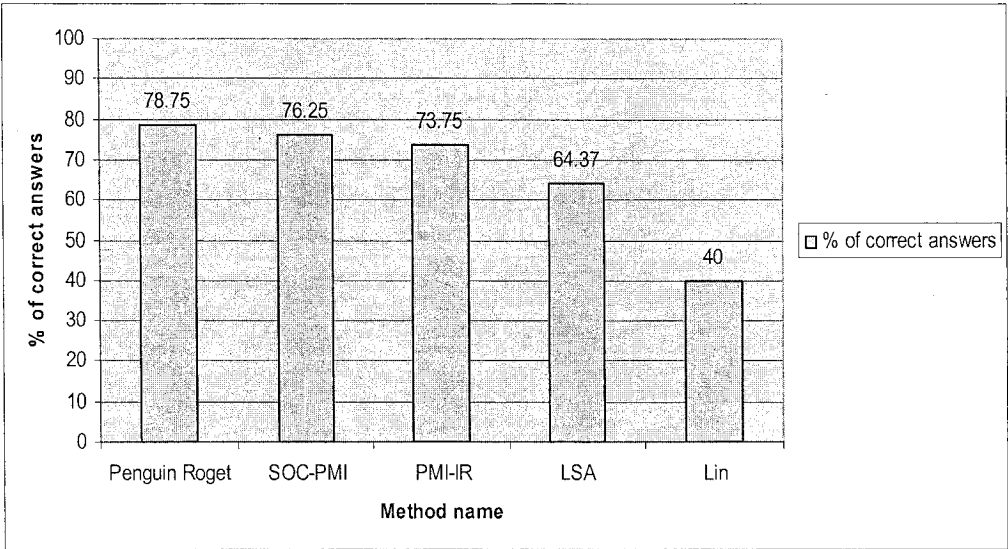


Figure 2.1: Results on the 80 TOEFL questions

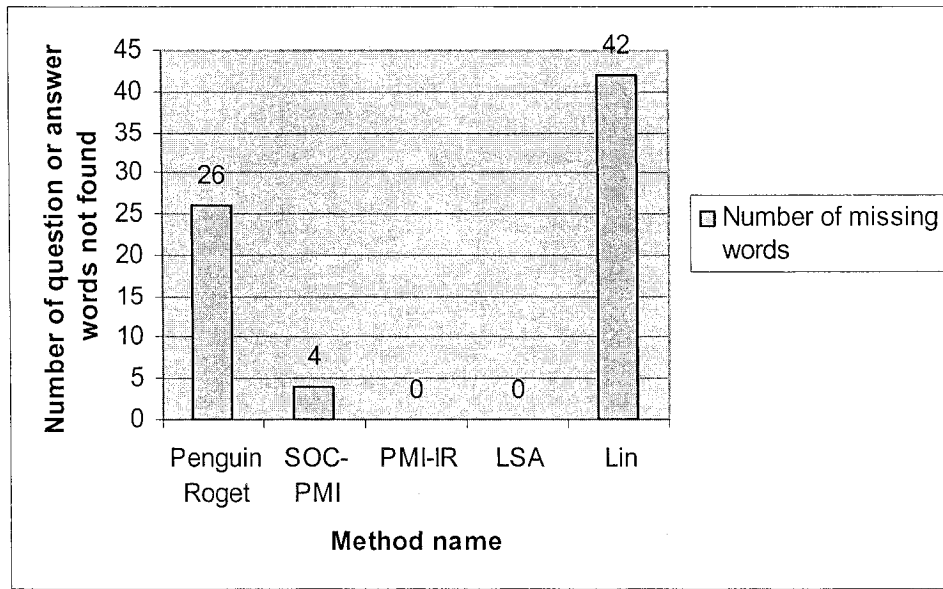


Figure 2.2: Number of missing words on the 80 TOEFL questions

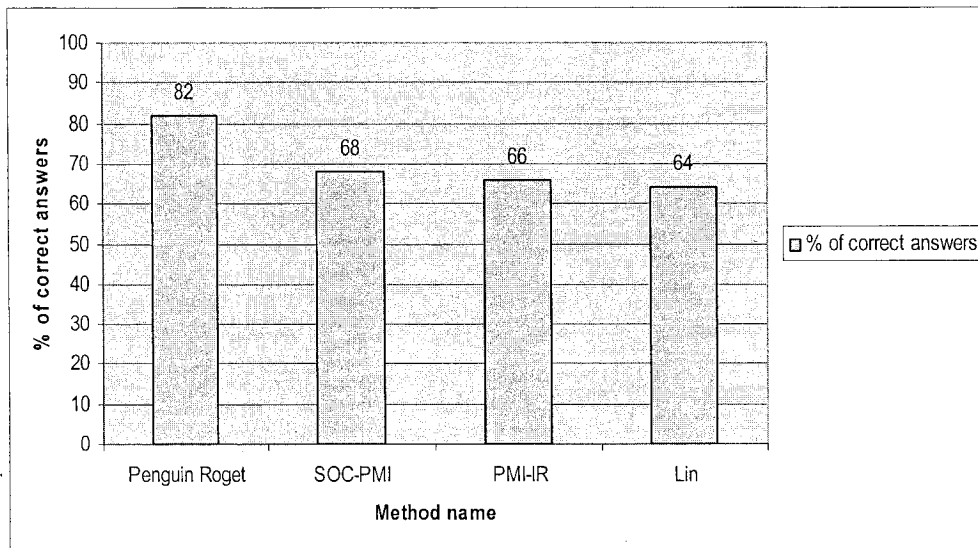


Figure 2.3: Results on the 50 ESL questions

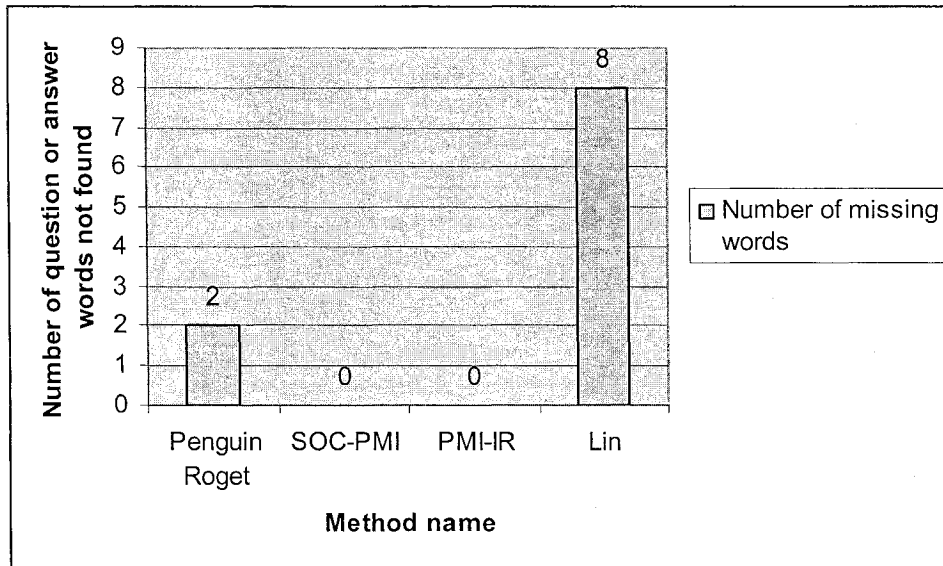


Figure 2.4: Number of missing words on the 50 ESL questions

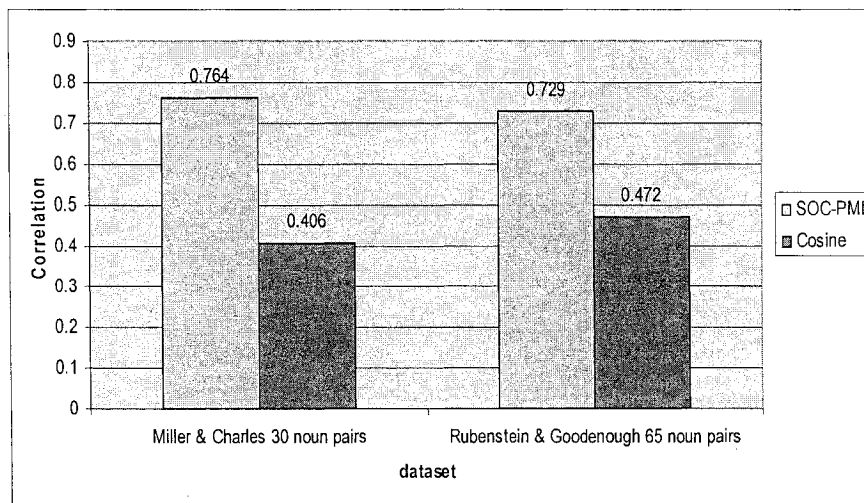


Figure 2.5: Correlation of noun pairs

For the 50 ESL questions, the SOC-PMI method correctly answered 68% of the questions (without using the context) compared to (Turney, 2001) where the best result was 66%, as shown in Figure 2.3. The number of question or answer words that different methods did not find for the 80 TOEFL questions and 50 ESL questions are in Figure 2.2

and Figure 2.4, respectively. These questions were not answered in the experiments because the similarity could not be computed for all pairs of words.

For Miller and Charles' (1991) dataset, we got a correlation of 0.764 with the human judges (Figure 2.5). For Rubenstein and Goodenough's (1965) dataset we got a correlation of 0.729. These correlation values are very good for a corpus-based measure, considering that a baseline vector space method using cosine obtains 0.406 for the first set and 0.472 for the second set. For dictionary-based measures (Jarmasz and Szpakowicz, 2003), the correlations are slightly higher, but comparable to ours.

Figure 2.1 and Figure 2.3 show that a method using Roget's thesaurus provides 2.5% and 14% more correct results than ours for the 80 TOEFL questions and 50 ESL questions, respectively. The WordNet-based measures – implemented in the WordNet::Similarity package by Pedersen et al. (2004) – achieve lower accuracy on the two data sets than the Roget measure (Jarmasz and Szpakowicz, 2003). The fact that the Roget measure performs better than the corpus-based measures is to be expected, because Roget's thesaurus can be seen as a classification system. It is composed of six primary classes and each is composed of multiple divisions and then sections. This may be conceptualized as a tree containing over a thousand branches for individual *meaning clusters* or semantically linked words. Though these words are not exactly synonyms, but can be viewed as colors or connotations of a meaning or as a spectrum of a concept. One of the most general words is chosen to typify the spectrum as its headword, which labels the whole group.

Second Order Co-occurrence PMI may be helpful as a tool to aid in the automatic construction of the synonyms of a word. This could be a future addition to our proposed method.

Detecting semantic outliers in speech recognition transcripts can use semantic similarity measures (Inkpen and Désilets, 2005) and a corpus-based similarity measures played an important role because of its large type coverage. The corpus-based measure was shown to perform better than the Roget-based measure in the task. In future work we can try the SOC-PMI method for this task.

2.6 Summary

This chapter presents a new corpus-based method for calculating the semantic similarity of two target words. Our method, called Second Order Co-occurrence PMI (SOC-PMI), uses Pointwise Mutual Information to sort lists of important neighbor words of the two target words. Then we consider the words which are common in both lists and aggregate their PMI values (from the opposite list) to calculate the relative semantic similarity. Our method was empirically evaluated using Miller and Charles's (1991) 30 noun pair subset, Rubenstein and Goodenough's (1965) 65 noun pairs, 80 synonym test questions from the Test of English as a Foreign Language (TOEFL), and 50 synonym test questions from a collection of English as a Second Language (ESL) tests. Evaluation results show that our method outperforms several competing corpus-based methods.

Chapter 3

Corpus-based Word Segmentation

In this chapter we present a new corpus-based method for automatic word segmentation. The method formulates a generalized approach to word segmentation using maximum-length descending-frequency and entropy rate. We will also use this method in database schema matching in Chapter 4.

3.1 Word Segmentation

Word segmentation is an important problem in many natural language processing tasks; for example, in speech recognition where there is no explicit word boundary information given within a continuous speech utterance, or in interpreting written languages such as Chinese, Japanese and Thai where words are not delimited by white-space but instead must be inferred from the basic character sequence. We differentiate the terms *word breaking* and *word segmentation*. Word breaking refers to the process of segmenting known words that are predefined in a lexicon. Word segmentation refers to the process of both lexicon word segmentation and unknown word or new word³ detection. Automatic word segmentation is a basic requirement for unsupervised learning in morphological analysis. Developing a morphological analyzer for a new language by hand can be costly and time consuming, requiring a great deal of effort by highly-specialized experts.

³ New words in this thesis paper refer to out-of-vocabulary words that are neither recognized as named entities or factoids, nor derived by morphological rules. These words are mostly domain-specific and / or time-sensitive.

In databases, word segmentation can be used in schema matching to solve semantic heterogeneity, a key problem in any data sharing system whether it is a federated database, a data integration system, a message passing system, a web service, or a peer-to-peer data management system (Madhavan et al., 2005). The name of an element in a database typically contains words that are descriptive of the element's semantics. N -grams⁴ have been shown to work well in the presence of short forms, incomplete names and spelling errors that are common in schema names (Do and Rahm, 2002).

Also, extracting words (word segmentation) from a scanned document page or a PDF is an important and basic step in document structure analysis and understanding systems; incorrect word segmentation during OCR leads to errors in information retrieval and in understanding the document.

One of the common approaches involving an extensive word list combined with an informed segmentation algorithm can help achieve a certain degree of accuracy in word segmentation, but the greatest barrier to accurate word segmentation is in recognizing unknown words, words not in the lexicon of the segmenter. This problem is dependent both on the source of the lexicon as well as the correspondence between the text in question and the lexicon. Fung and Wu (1994) reported that segmentation accuracy is significantly higher when the lexicon is constructed using the same type of corpus as the corpus on which it is tested.

The term *maximum-length descending-frequency* means that we choose maximum length n -grams that have a minimum threshold frequency and then we look for further n -grams in descending order based on length. If two n -grams have same length then we choose the n -gram with higher frequency first and then the n -gram with next higher frequency if any of its characters are not a part of the previous one. If we follow this procedure, after some iterations, we can be in a state with some remaining character(s) (we call it *residue*) that is not matched with any type in the corpus. To solve this, we use the *leftMaxMatching* and *rightMaxMatching* algorithms presented in Section 3.3 along with entropy rate.

⁴ Sequence of n consecutive characters

3.2 Related Work

Word segmentation methods can be roughly classified as either dictionary-based or statistically-based methods, while many state-of-the-art systems use hybrid approaches. In dictionary-based methods, given an input character string, only words that are stored in the dictionary can be identified. The performance of these methods thus depends to a large degree upon the coverage of the dictionary, which unfortunately may never be complete because new words appear constantly. Therefore, in addition to the dictionary, many systems also contain special components for unknown word identification. In particular, statistical methods have been widely applied because they use a probabilistic or cost-based scoring mechanism rather than a dictionary to segment the text (Gao et al., 2005).

A simple word segmentation algorithm is to consider each character a distinct word. This is practical for Chinese because the average word length is very short, usually between one and two characters, depending on the corpus (Fung and Wu, 1994), and actual words can be recognized with this algorithm. Although it does not assist in task such as parsing, part-of-speech tagging, or text-to-speech systems (Sproat et al., 1994), the character-as-word segmentation algorithm has been used to obtain good performance in Chinese information retrieval, a task in which the words in a text play a major role in indexing.

One of the most popular methods is *maximum matching* (MM), usually augmented with heuristics to deal with ambiguities in segmentation. Another very common approach to word segmentation is to use a variation of the *maximum matching algorithm*, frequently referred to as the *greedy algorithm*. The greedy algorithm starts at the first character in a text and, using a word list for the language being segmented, attempts to find the longest word in the list starting with that character. If a word is found, the maximum-matching algorithm marks a boundary at the end of the longest word, then begins the same longest match search starting at the character following the match. If no match is found in the word list, the greedy algorithm simply segments that character as a word and begins the search starting at the next character. A variation of the greedy algorithm segments a sequence of unmatched characters as a single word; this variant is more likely to be successful in writing systems with longer average word lengths. In this

manner, an initial segmentation can be obtained that is more informed than a simple character-as-word approach. As a demonstration of the application of the character-as-word and greedy algorithms, consider an example of “desegmented” English, in which all the white space has been removed: the “desegmented” version of the text, *the most favourite music of all time*, would thus be *themoستfavourite music of alltime*. Applying the character-as-word algorithm would result in the useless sequence of tokens *themoستfavourite music of alltime*, which is why this algorithm only makes sense for languages such as Chinese. Applying the greedy algorithm with a “perfect” word list containing all known English words would first identify the word *them*, since that is the longest sequence of letters starting at the initial *t* which forms an actual word. Starting at the *o* following *them*, the algorithm would then find no match. Continuing in this manner, *themoستfavourite music of alltime* would be segmented by the greedy algorithm as *them o st favourite music of all time*. A variant of the maximum matching algorithm is the *reverse maximum matching* algorithm, in which the matching proceeds from the end of the string of characters, rather than the beginning. In the foregoing example, *themoستfavourite music of alltime* would be segmented as *the most favourite music o fall time* by the reverse maximum matching algorithm. Greedy matching from the beginning and the end of the string of characters enables an algorithm such as *forward-backward matching*, in which the results are composed and the segmentation optimized based on the two results (Dale et al., 2000).

Many unsupervised methods have been proposed for segmenting raw character sequences with no boundary information into words (Brent and Cartwright, 1996; Brent, 1999; Deligne and Bimbot, 1995; Christiansen, 1997; Christiansen et al., 1998; Kit and Wilks, 1999; Hua, 2000). Brent (1999) gives a good survey of these methods. Most current approaches are using some form of Expectation Maximization (EM) to learn a probabilistic speech-or-text model and then employing Viterbi decoding procedures (Rabiner, 1989) to segment new speech or text into words. One reason that EM is widely adopted for unsupervised learning is that it is guaranteed to converge to a good probability model that locally maximizes the likelihood or posterior probability of the training data. For the problem of word segmentation, EM is typically applied by first extracting a set of candidate multi-grams from a given training corpus (Deligne and

Bimbot, 1995), initializing a probability distribution over this set, and then using the standard iteration to adjust the probabilities of the multi-grams to increase the posterior probability of the training data.

Saffran et al., (1996) proposed that word segmentation from continuous speech may be achieved by using transitional probabilities (TP) between adjacent syllables A and B , where, $TP(A \rightarrow B) = P(AB)/P(A)$, with $P(AB)$ being the frequency of B following A , and $P(A)$ the total frequency of A . Word boundaries are postulated at local minima, where the TP is lower than its neighbors.

In corpus-based word segmentation, there is either no explicit model learnt, as when neural networks (Rumelhart and McClelland, 1986) or lazy learning (Daelamans et al., 1997) are used, or the derived models are less sophisticated and do not use any abstractions of the word constituents found in data (Brill, 1994; Mikheev, 1997). Using annotated corpora greatly facilitates learning. However, there are situations in which one is interested in Unsupervised Learning (UL), that is, from unannotated corpora. Motivation for UL can vary from purely pragmatic, such as the high cost or unavailability of annotated corpora, to theoretical, when language is modelled as yet another communication code within the framework of Information Theory (Shannon and Weaver, 1963).

3.3 Proposed Word Segmentation method

Let $S = l_1 l_2 l_3 \dots l_m$ denotes a text of m consecutive characters without any space in between them for which we need to segment and $C = \{c_1, c_2, \dots, c_\tau\}$ denotes a large corpus of text containing τ words (tokens). Also, let $T^p = \{t_1, t_2, \dots, t_p\}$ be the set of all (p) unique words (types) which occur in the corpus C and $T^f = \{f_1, f_2, \dots, f_p\}$ be the set of frequencies of all the corresponding types in T^p i.e. f_x is the frequency of type t_x . Unlike the corpus C , which is an ordered list containing many occurrences of the same words, T^p is a set containing no repeated words. Again, let n is the maximum length of any possible words in the segmented words list where $n \leq m$ and $N^p = \{l_1, l_2, \dots, l_n, l_1 l_2, l_2 l_3, \dots, l_1 l_2 \dots l_n, \dots\}$ be the set of all possible n -grams where $\eta = |N^p|$ is the total number of n -grams in N^p . We can also consider N^p as $N^p = \{w_1, w_2, \dots, w_\eta\}$. And $N^f = \{f'_1, f'_2, \dots, f'_\eta\}$ be the set of frequencies of all the corresponding n -grams of N^p taken from T^f , i.e. f'_x is the frequency

of w_x . To get rid of the noise types of the corpus, we assign a set of minimum frequencies for each possible length from 1 to n to be considered as a valid word. $M^f = \{\alpha_1, \alpha_2, \dots, \alpha_n\}$, where α_x is the minimum frequency required to be a valid word of length x . Minimum required frequency, α_x is inversely proportional to the word length, x . The steps of the method are as follows:

Step 1: Sort all the elements of N^p in descending order based on length (in characters). Again sort in descending order for same length words of the sorted N^p (say $\overline{N^p}$) based on the frequencies of N^f . For each element in $\overline{N^p}$ do the next steps:

Step 2: If $S \neq \emptyset$ and the current maximum length n -gram (say w_n) in $\overline{N^p}$ satisfies $f_n \geq \alpha_{|w_n|}$ and $w_n \in S$ (i.e., $S \cap w_n = w_n$) then add w_n to segmented word list, S' (i.e., $S' \leftarrow S' \cup w_n$) and remove w_n from S (i.e., $S \leftarrow S \setminus w_n$) and add a blank space as a boundary mark.

Step 3: If $S \neq \emptyset$ and not all elements in $\overline{N^p}$ are done then update w_n by the next maximum length n -gram from $\overline{N^p}$ and go to step 2.

Step 4: Rearrange all the words of S' in accordance with S . If $S = \emptyset$, then output S' and exit. Otherwise, for each remaining chunks⁵, r in S call *matchResidue*(r), output S' and exit.

⁵ A single chunk may contain one or more characters.

Algorithm *matchResidue*

Input: r, S' // r is the remaining chunk and S' is the current segmented word list

1. // Take the prefix word, w_{n-1} and suffix word, w_n of r from S'
2. // according to the would be position of r in S' .
3. $S' \leftarrow S' \setminus w_{n-1}$
4. $S' \leftarrow S' \setminus w_n$
5. $S_t \leftarrow w_{n-1} \cup r \cup w_n$
6. // $S_t = \{l_1 l_2 l_3 \dots l_m\}$, where m is the length of S_t
7. $S_t' \leftarrow \text{leftMaxMatching}(S_t)$
8. $S_t'' \leftarrow \text{rightMaxMatching}(S_t)$
9. **if** ($|S_t'| > |S_t''|$)
10. $S' \leftarrow S' \cup S_t''$
11. **elseif** ($|S_t'| < |S_t''|$)
12. $S' \leftarrow S' \cup S_t'$
13. **else**
14. find a $x \in \{S_t', S_t''\}$ for which entropy rate $\frac{1}{|x|} \sum_{i=1}^{|x|} \log_2(f_i)$ is maximum
15. $S' \leftarrow S' \cup x$
16. **end**

Output: S' // S' is the segmented word list after matching residue

Figure 3.1: Matching residue

Algorithm *leftMaxMatching*

// n is the maximum length of any possible valid words in S_i and $n \leq m$

Input: S_i // S_i is a “desegmented” word

1. **while** $S_i \neq \emptyset$ **do**
2. $N^p \leftarrow \{l_1, l_1l_2, l_1l_2l_3 \dots, l_1l_2 \dots l_n\}$
3. i.e., $N^p \leftarrow \{w_1, w_2 \dots, w_n\}$
4. $N^f \leftarrow \{f_1', f_2', \dots, f_n'\}$
5. $M^f \leftarrow \{a_1, a_2, \dots, a_n\}$
6. $i \leftarrow 1$
7. **while** ($i \leq n \ \&\& \ i \leq m$)
8. **if** ($f_i' \geq a_i$)
9. $max \leftarrow i$
10. **end**
11. increment i
12. **end**
13. $S_i' \leftarrow S_i' \cup w_{max}$
14. $S_i \leftarrow S_i \setminus w_{max}$
15. **end**

Output: S_i' // S_i' is the segmented word list after leftmax matching

Figure 3.2: Matching leftmax

Algorithm *rightMaxMatching*

// n is the maximum length of any possible valid words in S_t and $n \leq m$

Input: S_t // S_t is a “desegmented” word

1. **while** $S_t \neq \emptyset$ **do**
2. $N^p \leftarrow \{l_m, l_{m-1}l_m, l_{m-2}l_{m-1}l_m, \dots, l_{m-n}l_{m-n+1} \dots l_m\}$
3. i.e., $N^p \leftarrow \{w_1, w_2, \dots, w_n\}$
4. $N^f \leftarrow \{f'_1, f'_2, \dots, f'_n\}$
5. $M^f \leftarrow \{\alpha_1, \alpha_2, \dots, \alpha_n\}$
6. $i \leftarrow 1$
7. **while** ($i \leq n \ \&\& \ i \leq m$)
8. **if** ($f'_i \geq \alpha_i$)
9. $max \leftarrow i$
10. **end**
11. increment i
12. **end**
13. $S'_t \leftarrow S_t \cup w_{max}$
14. $S_t \leftarrow S_t \setminus w_{max}$
15. **end**

Output: S'_t // S'_t is the segmented word list after rightmax matching

Figure 3.3: Matching rightmax

In *matchResidue*, if *leftMaxMatching* and *rightMaxMatching* return same numbers of words then we use entropy rate to decide which set of words we will accept. The intuition behind using entropy rate is that if we have a set of words having larger average frequency (we use normalized frequency in the entropy rate) than the other set of words, it is obvious that the first set of words is more meaningful than the second set of words.

3.4 A Walk-Through Example

As a demonstration of the application of the proposed algorithms, consider the same example of “desegmented” English text, $S = \textit{themostfavouritemusicofalltime}$. We have

used the BNC⁶ (British National Corpus) to calculate T^p and T^f . let, $n=10$ is the maximum length⁷ of all possible words in S and $M^f = \{1000, 500, 50, 16, 15, 12, 10, 3, 2, 2\}$. Table 3.1 shows the sorted n -grams, $\overline{N^p}$ and their frequencies, N^f for this specific example.

For each element w_n (say, *favourite*) in $\overline{N^p}$,

Step 2: w_n satisfies $f_n \geq \alpha_{|w_n|}$ as $4671 \geq 2$ and w_n is a substring of S .

$S' = \{favourite\}$ and $S = \{themost musicofalltime\}$.

Step 3: Not all elements in $\overline{N^p}$ are done, update $w_n = \{alltime\}$ and go to step 2.

Step 2: doesn't satisfy $f_n \geq \alpha_{|w_n|}$ as $6 < 10$ though w_n is a substring of S .

Step 3: Not all elements in $\overline{N^p}$ are done, update $w_n = \{favour\}$ and go to step 2.

Step 2: Condition fails as w_n is not a substring of S .

Step 3: Not all elements in $\overline{N^p}$ are done, update $w_n = \{musico\}$ and go to step 2.

$\overline{N^p}$	N^f	$\overline{N^p}$	N^f
favourite	4671	tfa	2
alltime	6	of	3052752
favour	6805	it	1054552
musico	10	he	641236
music	15134	me	131869
vouri	1	us	80206
them	167457	co	17476
time	164294	th	16486
most	98276	st	15565
fall	11202	al	7299
item	3780	fa	2172
rite	293	em	1641
allt	28	os	1005
emus	14	te	831

⁶ <http://www.natcorp.ox.ac.uk/>

⁷ Though in BNC, the length of the longest valid word is 34.

musi	3	si	658
hemo	3	Mo	639
emos	2	ti	615
the	6057315	im	576
all	282012	lt	485
our	93463	av	291
tim	3401	mu	276
hem	305	ll	233
sic	292	ri	230
mus	269	ou	151
emu	247	ic	132
ico	95	vo	93
uri	46	ur	77
fal	44	tf	11
ofa	36	a	2179299
mos	36	i	873059
fav	33	l	59
tem	31	c	46
emo	20	t	21
ost	18	s	19
rit	13	e	17
ite	11	r	14
usi	8	h	12
ime	6	f	10
cof	5	v	9
avo	5	m	8
lti	4	o	5
vou	3	u	3

Table 3.1: Sorted n -grams and their frequencies

Step 2: Condition fails as w_n does not satisfy $f_n' \geq \alpha_{|w_n|}$ as $10 < 12$.

Step 3: Not all elements in $\overline{N^p}$ are done, update $w_n = \{music\}$ and go to step 2.

Step 2: w_n satisfies $f_n' \geq \alpha_{|w_n|}$ as $15134 \geq 15$ and w_n is a substring of S .

$S' = \{favourite, music\}$ and

$S = \{themost ofalltime\}$.

We will only show the step 2 of all the remaining elements in $\overline{N^p}$ that satisfy the conditions.

Step 2: $w_n = \{them\}$

$S' = \{favourite, music, them\}$ and $S = \{ost ofalltime\}$.

Step 2: $w_n = \{time\}$

$S' = \{favourite, music, them, time\}$ and

$S = \{ost ofall\}$.

Step 2: $w_n = \{fall\}$

$S' = \{favourite, music, them, time, fall\}$ and

$S = \{ost o\}$.

Step 4: Rearrange $S' = \{them, favourite, music, fall, time\}$ and $S \neq \emptyset$, so call *matchResidue*(*ost*) and then *matchResidue*(*o*).

Case 1: *matchResidue*(*ost*) is called

$S' = S' \setminus \{w_{n-1}, w_n\}$

$S' = \{them, favourite, music, fall, time\} \setminus \{them, favourite\}$

$= \{music, fall, time\}$

$S_t = \{themostfavourite\}$

$S_t' = \{them, os, t, favourite\} \leftarrow \mathbf{leftMaxMatching}(themostfavourite)$

$S_t'' = \{the, most, favourite\} \leftarrow \mathbf{rightMaxMatching}(themostfavourite)$

As $|S_t'| > |S_t''|$, $S' = \{music, fall, time\} \cup S_t''$

i.e., $S' = \{the, most, favourite, music, fall, time\}$

Case 2: *matchResidue*(*o*) is called

$S' = S' \setminus \{w_{n-1}, w_n\}$

$$S' = \{the, most, favourite, music, fall, time\} \setminus \{music, fall\}$$

$$= \{the, most, favourite, time\}$$

$$S_t = \{musicofall\}$$

$$S'_t = \{music, of, all\} \leftarrow \text{leftMaxMatching}(musicofall)$$

$$S''_t = \{mus, ico, fall\} \leftarrow \text{rightMaxMatching}(musicofall)$$

As in this case $|S'_t| = |S''_t|$, we need to find whether S'_t or S''_t maximizes the entropy rate,

$$\frac{1}{|x|} \sum_{i=1}^{|x|} \log_2(f_i), \text{ where } x \in \{S'_t, S''_t\}. \text{ The entropy rate for } S'_t \text{ is } (13.89 + 21.54 + 18.11) / 3$$

and for S''_t is

$$(8.07 + 6.57 + 13.45) / 3. \text{ So, } S' = \{the, most, favourite, time\} \cup S'_t, \text{ as } \frac{1}{|S'_t|} \sum_{i=1}^{|S'_t|} \log_2(f_i) >$$

$$\frac{1}{|S''_t|} \sum_{i=1}^{|S''_t|} \log_2(f_i). \text{ Finally, } S' = \{the, most, favourite, music, of, all, time\}.$$

3.5 Evaluation and Experimental Results

An obstacle to high-accuracy word segmentation is that there are no widely accepted guidelines for what constitutes a word; therefore, there is no agreement on how to “correctly” segment a text in a “desegmented” language. Native speakers of a language do not always agree about the “correct” segmentation, and the same text could be segmented into several very different (and equally correct) sets of words by different native speakers. Such ambiguity in the definition of what constitutes a word makes it difficult to evaluate segmentation algorithms that follow different conventions, as it is nearly impossible to construct a “gold standard” against which to directly compare results (Dale et al., 2000). As shown in Sproat et al. (1996), the rate of agreement between two human judges on this task is less than 80%.

The performance of word segmentation is usually measured using *precision* and *recall*, where recall is defined as the percentage of words in the manually segmented text identified by the segmentation algorithm and precision is defined as the percentage of words returned by the algorithm that also occurred in the hand-segmented text in the same position. In general, it is easy to obtain high performance for one of the two measures but relatively difficult to obtain high performance for both. *F-measure* (F) is

the geometric mean of *precision* (P) and *recall* (R) and expresses a trade-off between those two measures. These performance measures are defined as follows:

$$P = TP / (TP + FP)$$

$$R = TP / (TP + FN)$$

$$F = (1 + \beta)PR / (\beta P + R)$$

= $2PR / (P + R)$, with $\beta = 1$ such that *precision* and *recall* weighted equally. Here, TP , FP and FN stand for True Positive, False Positive and False Negative respectively.

For instance, if the target segmentation is “we are human”, and the model outputs “weare human”, then precision is 1/2 (“human” out of “weare” and “human”, recall is 1/3 (“human” out of “we”, “are”, and “human”) and F-measure is 2/5.

We used the type frequency from BNC and tested our segmentation method on part of the BNC corpus. Specifically, we converted a portion of the corpus to lowercase letters and removed all white space and punctuation. We used 285K characters and 57904 tokens as our test data. We obtained 84.28% word precision rate, 81.63% word recall rate, and 82.93% word F-measure.

In a second test, we used the type frequency from BNC and tested our segmentation method on the Brown corpus to make sure that we test on different vocabulary from the training data. This insures that some of the word in the test set were not previously seen (out-of-vocabulary words). There were 4,705,022 characters and 1,003,881 tokens in the Brown corpus. We obtained 89.92% word precision rate, 94.69% word recall rate, and 92.24% word F-measure. The average number of tokens per line could be the reason for obtaining better result when we tested on the Brown corpus, as 8.49 and 16.07 are the average number of tokens per line in the Brown corpus and the BNC corpus, respectively.

One of the best known results on segmenting the Brown corpus is due to Kit and Wilks (1999) who use a description-length gain method. They trained their model on the whole corpus (6.13M) and reported results on the *training* set, obtaining a boundary precision of 79.33%, a boundary recall of 63.01% and boundary F-measure of 70.23%. Peng and Schuurmans (2001) trained their model on a subset of the corpus (4292K) and tested on *unseen* data. After the lexicon is optimized, they obtained 16.19% higher recall and 4.73% lower precision; resulting in an improvement of 5.2% in boundary F-measure. De Marcken (1995) also used a minimum description length (MDL) framework and a

hierarchical model to learn a word lexicon from raw speech. However, this work does not explicitly yield word boundaries, but instead recursively decomposes an input string down to the level of individual characters. As pointed out by Brent (1999), this study gives credit for detecting a word if any node in the hierarchical decomposition spans the word. Under this measure (de Marcken, 1995) reports a word recall rate of 90.5% on the Brown corpus. However, his method creates numerous chunks and therefore only achieves a word precision rate of 17%. Christiansen et al. (1998) used a simple recurrent neural network approach and report a word precision rate of 42.7% and word recall rate of 44.9% on spontaneous child-directed British English. Brent and Cartwright (1996) used a MDL approach and reported a word precision rate of 41.3% and a word recall rate of 47.3% on the CHILDES collection. Brent (1999) achieved about 70% word precision and 70% word recall by employing additional language modeling and smoothing techniques. Peng and Schuurmans (2001) obtained 74.6% word precision rate, 79.2% word recall rate, and 75.4% word F-measure on the Brown corpus. A balance of high precision and high recall is the main advantage of our proposed method. However, it is difficult to draw a direct comparison between these results because of the different test corpora used by different authors. Figure 3.4 summarizes the result of different methods which are tested on the Brown corpus based on precision, recall and F-measure. Though all the methods in Figure 3.4 use the Brown corpus, the testing data sets in the Brown corpus are not exactly the same.

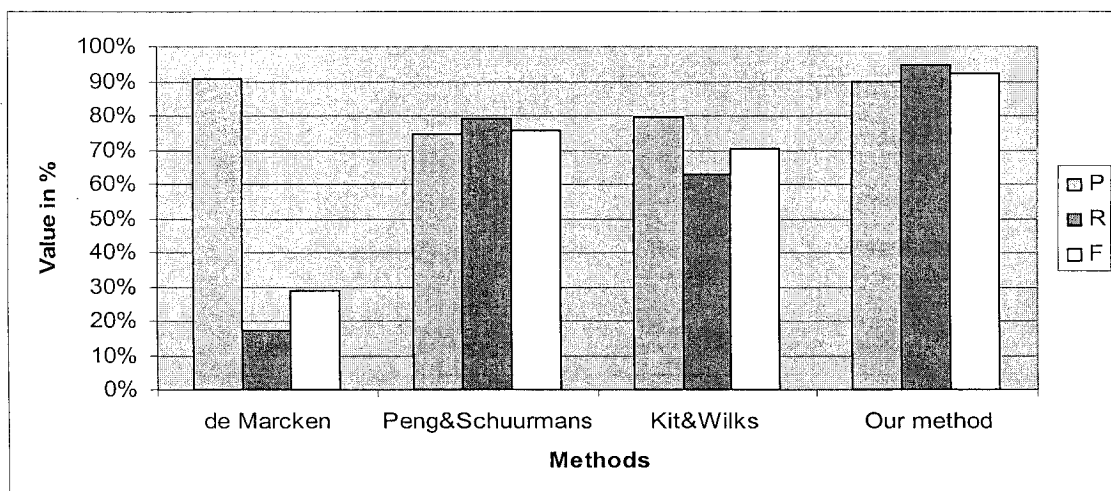


Figure 3.4 : Test result on the Brown corpus

Actually, this method can effectively distill new words, special terms and proper nouns when the corpus covers a huge collection of both domain-dependent and domain-independent words, and it can effectively avoid statistical errors in shorter strings which belong to a longer one. However, names are not always easy to exploit and contain abbreviations and special characters that vary between domains. This method can be used to address this issue, an important step of schema matching in databases. A generalized characteristic of this method is that it can be extended as a dictionary-based method or hybrid method with some additions to the algorithms. The absence of type frequencies in dictionary means that we can only use the length of the types. In that case, we need to focus on what type to choose for the same length types that share some common characters. Again, we can not choose whether we take the elements of *leftMaxMatching* or *rightMaxMatching* when both of them return the same number of elements as we can not use entropy rate for the absence of type frequencies. Experimental results show that our method can segment words with high precision and high recall.

Top choices search engines segment the ‘desegmented’ part from a search text only if the ‘desegmented’ part contains two to three words. The popular search engine Google segments a ‘desegmented’ part of search text consisting of only two words and fails to provide any search result when the search text consists of more than two ‘desegmented’ words. For example, when we search for “the most favourite music of all time” by Google, we receive about 164 million web search results. When we google “themost favourite music of all time”, we receive a suggestion “Did you mean: *the most favourite music of all time*”. But when we google “themostfavourite music of all time”, we receive “Your search – **themostfavourite music of all time** - did not match any documents.” and the following four suggestions:

- Make sure all words are spelled correctly.
- Try different keywords.
- Try more general keywords.
- Try fewer keywords.

From the popular search engine of twenty first’s century, the user could expect a more rational suggestion like “Did you mean: *the most favourite music of all time*”. We test

the word segmentation performance of twelve search engines, with three different search texts and the result is listed in the following Table 3.2, where the symbol \sqrt means that the search engine succeed to segment the ‘desegmented’ words and \times means that the search engine fails to segment the ‘desegmented’ words.

ST_1 = “themost favourite music of all time”

ST_2 = “themostfavourite music of all time”

ST_3 = “themostfavouritemusic of all time”

The search texts ST_1 , ST_2 and ST_3 have two, three and four words in the ‘desegmented’ part respectively.

Search engine	Web address of the search engine	ST_1	ST_2	ST_3	Comments
Google	www.google.com	\sqrt	\times	\times	
Yahoo	www.yahoo.com	\sqrt	\sqrt	\times	
Ask Jeeves	www.askjeeves.com	\sqrt	\times	\times	
AllTheWeb.com	www.alltheweb.com	\sqrt	\sqrt	\times	Uses Yahoo! Search Technology
AOL Search	www.aol.com	\sqrt	\times	\times	
Hot Bot	www.hotbot.com	\sqrt	\times	\times	Uses Ask Jeeves or Google Technology
Teoma	www.teoma.com	\sqrt	\times	\times	Uses Ask Jeeves Technology
AltaVista	www.altavista.com	\sqrt	\sqrt	\times	
Gigablast	www.gigablast.com	\sqrt	\times	\times	
LookSmart	www.looksmart.com	\times	\times	\times	
Lycos	www.lycos.com	\sqrt	\times	\times	
MSN Search	www.search.msn.com	\sqrt	\times	\times	

Table 3.2: Word segmentation performance of twelve search engines

3.6 Summary

In this chapter, we formulated a generalized method of automatic word segmentation. The method uses corpus type frequency information to choose the type with maximum length and frequency from “desegmented” text. It also uses a modified forward-backward matching technique using maximum length frequency and entropy rate if any non-matching portions of the text exist. The method is extendable to a dictionary-based or hybrid method with some additions to the algorithms. Evaluation results show that our method outperforms several competing methods.

Chapter 4

Name-based Schema Matching in Databases

In this chapter we present a new Name-based Element-level Schema Matching Method for databases that uses the word similarity method described in Chapter 2 and the word segmentation method described in Chapter 3.

4.1 Schema Matching

Schema matching is the problem of identifying elements of two given schemas that correspond to each other. It has been the focus of research since the 1970s in the Artificial Intelligence, Databases, and Knowledge Representation communities. Schema matching can also be defined as discovering semantically corresponding attributes in different schemas. Traditionally, the problem of matching schemas has essentially relied on finding pairwise-attribute correspondences. Though Schema matching identifies elements that correspond to each other, it does not explain how they correspond. For example, it might say that FirstName and LastName in one schema are related to Name in the other, but it does not say that concatenating the former yields the latter. Automatically discovering these correspondences or matches is inherently difficult.

Today, many researchers realize that schema matching is a core problem in e-commerce exchanges, in data integration/warehousing, and in Semantic Web

applications. Schema matching is fundamental for enabling query mediation and data exchange across information sources (Seligman et. al., 2002; Batini et. al., 1986). While schema matching has always been a problematic and interesting aspect of information integration, the problem is exacerbated as the number of information sources to be integrated, and hence the number of integration problems that must be solved, grows. Such schema matching problems arise both in “classical” scenarios such as company mergers, and in “new” scenarios such as the integration of diverse sets of queryable information sources over the Web. Purely manual solutions to the schema matching problem are too labor intensive to be scalable; as a result, there has been a great deal of research into automated techniques that can speed up this process by either automatically discovering good mappings, or at least by proposing likely matches that are then verified by a human expert (Kang and Naughton, 2003).

Rahm and Bernstein (2001) point out that it is not possible to determine fully automatically all matches between two schemas, primarily because most schemas have some semantics that affects the matching criteria but is not formally expressed or often not even documented. The implementation of the matching should therefore only determine *match candidates*, which the user can accept, reject or change. Furthermore, the user should be able to specify matches for elements for which the system was unable to find satisfactory match candidates.

4.2 Classification of Schema Matching Approaches

Rahm and Bernstein (2001) classify the major approaches to schema matching. Figure 4.1 shows part of their classification together with some sample approaches. An implementation of matching may use multiple match algorithms or *matchers*.

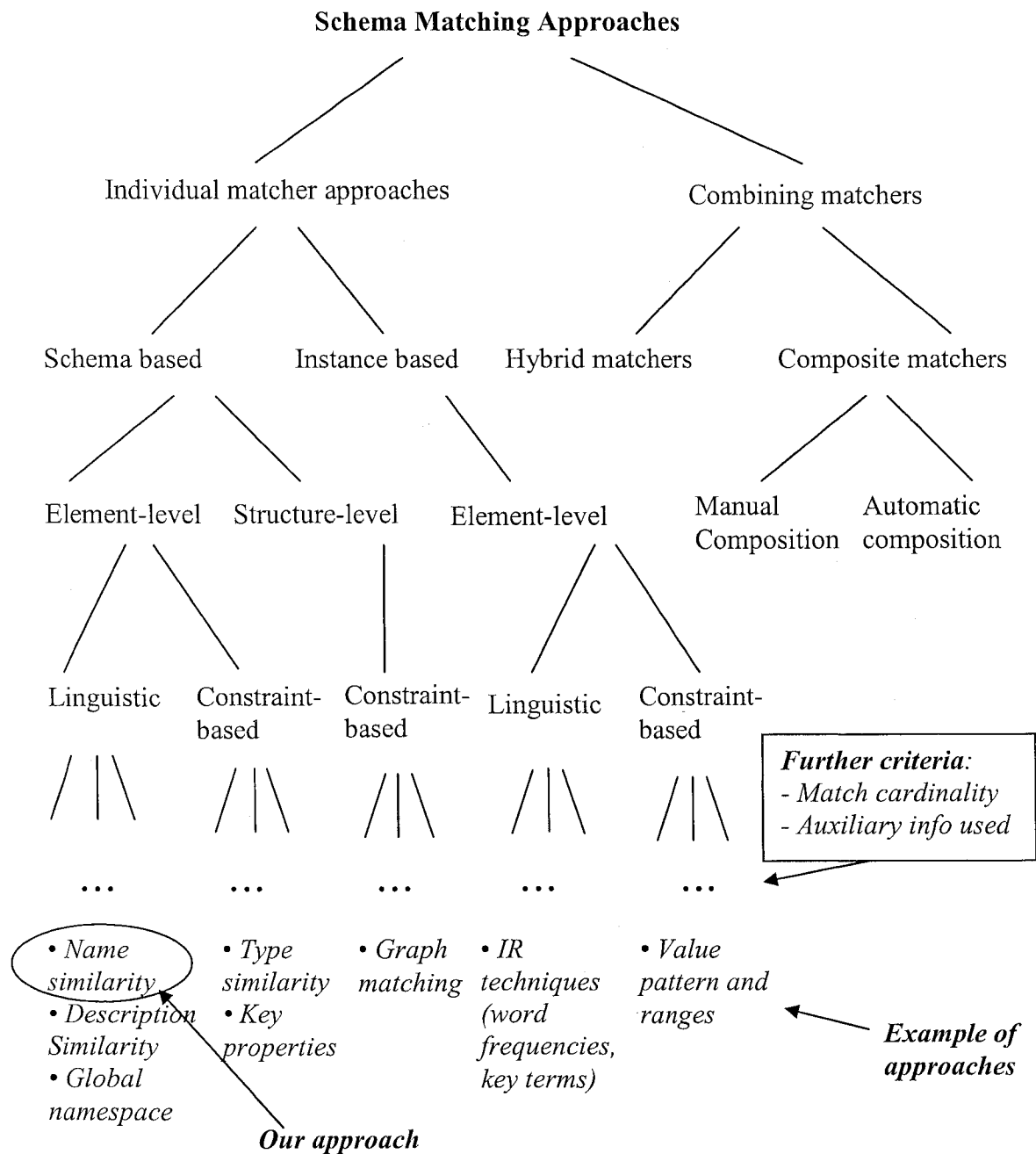


Figure 4.1: Classification of schema matching approaches (Rahm & Bernstein, 2001)

This allows selecting the matchers depending on the application domain and schema types. Given that we want to use multiple matchers we distinguish two sub-problems. First, there is the realization of individual matchers, each of which computes a mapping based on a single matching criterion. Second, there is the combination of individual matchers, either by using multiple matching criteria (e.g., name and type equality) within an integrated *hybrid matcher* or by combining multiple match results produced by different match algorithms within a *composite matcher*.

Linguistic Approaches

Linguistic matchers use element names and text (i.e., words/tokens or sentences) to find semantically similar schema elements. We discuss here two schema-level approaches, name matching and description matching.

Element Name Matching

Element name-based matching matches schema elements with equal or similar names. Similarity of names can be defined and measured in various ways, including:

- **Equality of name matching**

An important subcase is the equality of names from the same XML namespace, since this ensures that the same names indeed bear the same semantics.

- **Equality of canonical name representations after stemming and other preprocessing**

This is important to deal with special prefix/suffix symbols (e.g., CName → customer name, and EmpNO → employee number)

- **Equality of synonyms**

This is important to deal with synonyms (E.g., *car* synonymously matches *automobile*).

- **Equality of hypernyms**

Hypernym is a word that is more generic than a given word. For example, *fruit* is a hypernym of *apple* or *orange*. So, *apple* is-a *fruit* and *orange* is-a *fruit* imply *apple* hypernymically matches *fruit*, *orange* hypernymically matches *fruit*, and *apple* matches *orange* as siblings.

- **Similarity of names based on longest common substrings (LCS), edit distance, pronunciation, soundex** (a phonetic algorithm for indexing names by

their sound when pronounced in English and the basic aim is for names with the same pronunciation to be encoded into the same string so that matching can occur despite minor differences in spelling.), etc (Bell and Sethi, 2001). For example, *representedBy* matches *representative* based on common substrings. We experimented with these measures (edit, soundex, LCS etc.) thanks to a Perl package, produced by Grzegorz Kondrak from University of Alberta (Kondrak, 2005).

Solving any task related to synonyms and hypernyms normally requires the use of thesauri or dictionaries. General natural language dictionaries such as LDOCE⁸, Wordnet⁹ may be useful; perhaps even multi-language dictionaries (e.g., English-German or German-English) to deal with input schemas in different languages. In addition, name matching can use domain- or enterprise-specific dictionaries and *is-a* taxonomies containing common names, synonyms and descriptions of schema elements, abbreviations, etc. These specific dictionaries require a substantial effort to be built up in a consistent way. The effort may be worth the investment, especially for schemas with relatively flat structure where dictionaries provide the most valuable matching hints. But corpus-based methods could be a better choice than dictionary-based methods as a balanced corpus covers a huge collection of both domain-dependent and domain-independent words including special terms and proper nouns. Furthermore, tools are needed to enable names to be accessed and (re-)used, such as within a schema editor when defining new schemas.

Homonyms (one of two or more words that have the same sound and often the same spelling but differ in meaning) can mislead a matching algorithm as homonyms are similar names that refer to different elements. Clearly, homonyms may be a part of natural language, such as *bank* (embankment, river bank) and *bank* (place where money is kept). A name matcher can reduce the number of wrongly matched candidates by using mismatch information supplied by users or dictionaries, though it requires a substantial effort or at least, the matcher can offer a warning of the potential ambiguity due to multiple meanings of the name.

⁸ <http://www.longman.com/ldoce>

⁹ <http://wordnet.princeton.edu>

Name-based matching can identify multiple relevant matches for a given schema element i.e., it is not limited to finding just 1:1 matches. For example, it can match “address” with both “home address” and “office address”. In the case of synonyms and hypernyms, the join-like processing involves a list D of word pairs and their similarity as a further input. Assume a relation-like representation as follows:

S_1 (name₁₁, name₁₂ ...)

S_2 (name₂₁, name₂₂ ...)

D (name₁₁, name₂₁, similarity) where similarity is a similarity score for [name₁₁, name₂₁] between 0..1. Then a list of all match candidates can be generated by the following three-way join operation:

Select $S_1.name, S_2.name, D.similarity$

From S_1, S_2, D

Where ($S_1.name = D.name_{11}$) and

($D.name_{21} = S_2.name$) and

($D.similarity > threshold$)

The constraint here is that D will have to contain all relevant pairs of the transitive closure over similar names. For instance, if $\text{sim}(A, B) = 0.6$ and $\text{sim}(B, C) = 0.7$ are in D , then probabilistically we would expect D also to contain $\text{sim}(B, A) = 0.6$, $\text{sim}(C, B) = 0.7$, and $\text{sim}(A, C) = x$, $\text{sim}(C, A) = x$. Probabilistically, we would expect the similarity value x to be $0.6 \times 0.7 = 0.42$, but this depends on some factors such as the type of similarity, the use of homonyms, and perhaps other factors. For example, we might have $\text{sim}(\text{deliver}, \text{ship}) = 0.9$ and $\text{sim}(\text{ship}, \text{boat}) = 0.9$, but not $\text{sim}(\text{deliver}, \text{boat}) = x$ for a high similarity value x . Bright et. al. (1994) discuss another approach to assigning different weights to different types of similarity relationships.

Description matching

Often, schemas contain text descriptions of elements that typically explain the meaning of elements in natural language to express the intended semantics of schema elements. But the quality of these descriptions varies a lot. These comments can also be evaluated linguistically to determine the similarity between schema elements. For instance, this would help find that the following elements match, by a linguistic analysis of the comments associated with each schema element:

S_1 : empn // *employee name*

S_2 : name // *name of employee*

This linguistic analysis could be as simple as extracting keywords from the description which are used for synonym comparison, much like name matching.

Some approaches consider rule-based schema matching which are domain dependent (Milo and Zohar, 1998).

4.3 Proposed Name-based Element-level Schema Matching Method

We use *longest common subsequence* (LCS) (Allison, 1986) measure with some normalization and small modification for our string similarity measures. We use three different modified versions of LCS and then take weighted sum of these¹⁰. Kondrak (2005) showed that edit distance and the length of the longest common subsequence are special cases of n -gram distance and similarity, respectively. Melamed (1999) normalized LCS by dividing the length of the longest common subsequence by the length of the longer string and called it *longest common subsequence ratio* (LCSR). But LCSR does not take into account of the length of the smaller string which sometimes has a significant value in similarity score.

We normalize the *longest common subsequence* (LCS) so that it takes into account of the length of both the smaller and the larger strings and call it *normalized longest common subsequence* (NLCS) which is,

$$v_1 = NLCS(r_i, s_j) = \frac{\{length(LCS(r_i, s_j))\}^2}{length(r_i) \times length(s_j)}$$

While in classical LCS, the common subsequence need not to be consecutive, but in database schema matching, consecutive common subsequence is important for high degree of matching. We use *maximal consecutive longest common subsequence* starting at character 1, $MCLCS_1$ (Figure 4.2) and *maximal consecutive longest common subsequence* starting at any character n , $MCLCS_n$ (Figure 4.3). In Figure 4.2, we present an algorithm that takes two strings as input and returns the smaller string or maximal consecutive portions of the smaller string that consecutively match with the larger string,

¹⁰ We use modified versions because in our experiments we obtained better results (precision and recall for schema matching) than when using the original LCS, or other string similarity measures.

where matching must be from first character (character 1) for both of the strings. In Figure 4.3, we present another algorithm that takes two strings as input and returns the smaller string or maximal consecutive portions of the smaller string that consecutively match with the larger string, where matching may start from any character (character n) for both of the strings. We normalize $MCLCS_1$ and $MCLCS_n$ and call it *normalized MCLCS₁* ($NMCLCS_1$) and *normalized MCLCS_n* ($NMCLCS_n$) respectively.

$$v_2 = NMCLCS_1(r_i, s_j) = \frac{\{length(MCLCS_1(r_i, s_j))\}^2}{length(r_i) \times length(s_j)}$$

$$v_3 = NMCLCS_n(r_i, s_j) = \frac{\{length(MCLCS_n(r_i, s_j))\}^2}{length(r_i) \times length(s_j)}$$

We take the weighted sum of these individual v_1 , v_2 , and v_3 to determine string similarity score, where w_1 , w_2 , w_3 are weights and $w_1 + w_2 + w_3 = 1$. Therefore, the similarity of the two strings is:

$$\alpha = w_1 v_1 + w_2 v_2 + w_3 v_3.$$

We set equal weights for our experiments. Theoretically, $v_3 \geq v_2$.

For example, if $r_i = albastru$ and $s_j = alabaster$, then

$$LCS(r_i, s_j) = albastr$$

$$MCLCS_1(r_i, s_j) = al$$

$$MCLCS_n(r_i, s_j) = bast$$

$$NLCS(r_i, s_j) = 7^2 / (8 \times 9) = 0.68$$

$$NMCLCS_1 = 2^2 / (8 \times 9) = 0.056$$

$$NMCLCS_n(r_i, s_j) = 4^2 / (8 \times 9) = 0.22$$

$$\text{String similarity, } \alpha = w_1 v_1 + w_2 v_2 + w_3 v_3$$

$$= 0.33 \times 0.68 + 0.33 \times 0.056 + 0.33 \times 0.22 = 0.32$$

We then use word similarity measure, normalize it (Figure 4.4) and combine it with the string similarity to obtain a final similarity score. We now describe our schema matching method in detail.

Consider two given database schemas $R = \{R_1, R_2 \dots, R_\sigma\}$ and $S = \{S_1, S_2 \dots, S_\chi\}$; for each element in one database schema, we try to identify a matching element in the other schema, if any, using element names. We assume that schema R has σ elements and R_i is the element's name, where $i = 1 \dots \sigma$. Similarly, schema S has χ elements and S_j is the

element's name where $j = 1 \dots \chi$. Note that some elements in R can match multiple elements in S , and vice versa. So, our task is to identify whether an element name $R_i \in R$ matches an element name $S_j \in S$. Both R_i and S_j are strings of characters. Our method provides a similarity score between 0 and 1, inclusively. If the similarity score is above a certain threshold then the elements are considered as *match candidates*. If we set the threshold to 1 and the similarity score reaches this value, only then are we certain about their matching. For all other cases, we can only determine more or less probable *match candidates*. The method comprises the following six steps:

Step 1: We use all special characters, punctuations, and capital letters, if any, as initial word boundary and eliminate all these special characters and punctuations. After this initial word segmentation, we pass each of these segmented words to our word segmentation method and lemmatize to generate tokens. We assume $R_i = \{r_1, r_2 \dots, r_m\}$ has m tokens and $S_j = \{s_1, s_2 \dots, s_n\}$ has n tokens and $n \geq m$. Otherwise, we switch R_i and S_j .

Step 2: We count the number of r_i 's (say, δ) for which $r_i = s_j$, for all $r \in R_i$ and for all $s \in S_j$. I.e., there are δ tokens in R_i that exactly match with S_j , where $\delta \leq m$. We remove all δ tokens from both of R_i and S_j . So, $R_i = \{r_1, r_2 \dots, r_{m-\delta}\}$ and $S_j = \{s_1, s_2 \dots, s_{n-\delta}\}$. If $m-\delta = 0$, we go to step 6.

Step 3: We construct a $(m-\delta) \times (n-\delta)$ *matching matrix* (say, $M_1 = (\alpha_{ij})_{(m-\delta) \times (n-\delta)}$) using the following process: we assume any token $r_i \in R_i$ has τ characters, i.e., $r_i = \{c_1 c_2 \dots c_\tau\}$ and any token $s_j \in S_j$ has η characters, i.e., $s_j = \{c_1 c_2 \dots c_\eta\}$ where $\tau \leq \eta$. In other words, η is the length of the larger token and τ is the length of the smaller token. We calculate the followings:

$$\begin{aligned} v_1 &\leftarrow NLCS(r_i, s_j) \\ v_2 &\leftarrow NMCLCS_1(r_i, s_j) \\ v_3 &\leftarrow NMCLCS_n(r_i, s_j) \\ \alpha_{ij} &\leftarrow w_1 v_1 + w_2 v_2 + w_3 v_3 \end{aligned}$$

i.e., α_{ij} is a weighted sum of v_1 , v_2 , and v_3 where w_1 , w_2 , w_3 are weights and $w_1+w_2+w_3=1$.

We set equal weights for our experiments.

We put α_{ij} in row i and column j position of the matrix for all $i = 1 \dots m-\delta$ and $j = 1 \dots n-\delta$.

$$M_1 = \begin{bmatrix} \alpha_{11} & \alpha_{12} & \alpha_{1j} & \alpha_{1(n-\delta)} \\ \alpha_{21} & \alpha_{22} & \alpha_{2j} & \alpha_{2(n-\delta)} \\ \alpha_{i1} & \alpha_{i2} & \alpha_{ij} & \alpha_{i(n-\delta)} \\ \alpha_{(m-\delta)1} & \alpha_{(m-\delta)2} & \alpha_{(m-\delta)j} & \alpha_{(m-\delta)(n-\delta)} \end{bmatrix}$$

Step 4: We construct a $(m-\delta) \times (n-\delta)$ similarity matrix (say, $M_2 = (\beta_{ij})_{(m-\delta) \times (n-\delta)}$) using the following process: We put β_{ij} ($\beta_{ij} \leftarrow \text{similarityMatching}(r_i, s_j)$) (Figure 4.4)) in row i and column j position of the matrix for all $i = 1 \dots m-\delta$ and $j = 1 \dots n-\delta$.

$$M_2 = \begin{bmatrix} \beta_{11} & \beta_{12} & \beta_{1j} & \beta_{1(n-\delta)} \\ \beta_{21} & \beta_{22} & \beta_{2j} & \beta_{2(n-\delta)} \\ \beta_{i1} & \beta_{i2} & \beta_{ij} & \beta_{i(n-\delta)} \\ \beta_{(m-\delta)1} & \beta_{(m-\delta)2} & \beta_{(m-\delta)j} & \beta_{(m-\delta)(n-\delta)} \end{bmatrix}$$

Step 5: We construct another $(m-\delta) \times (n-\delta)$ joint matrix (say, $M = (\gamma_{ij})_{(m-\delta) \times (n-\delta)}$) using $M \leftarrow \psi M_1 + \phi M_2$ (i.e., $\gamma_{ij} = \psi \alpha_{ij} + \phi \beta_{ij}$) where ψ is the matching matrix weight factor, ϕ is the similarity matrix weight factor, and $\psi + \phi = 1$. Setting any one of these factors to 0 means that we do not include that matrix. Setting both of the factors to 0.5 means we consider them equally important.

$$M = \begin{bmatrix} \gamma_{11} & \gamma_{12} & \gamma_{1j} & \gamma_{1(n-\delta)} \\ \gamma_{21} & \gamma_{22} & \gamma_{2j} & \gamma_{2(n-\delta)} \\ \gamma_{i1} & \gamma_{i2} & \gamma_{ij} & \gamma_{i(n-\delta)} \\ \gamma_{(m-\delta)1} & \gamma_{(m-\delta)2} & \gamma_{(m-\delta)j} & \gamma_{(m-\delta)(n-\delta)} \end{bmatrix}$$

After constructing the *joint matrix*, M , we find out the maximum-valued matrix-element, γ_{ij} . We add this matrix element to a list (say, ρ and $\rho \leftarrow \rho \cup \gamma_{ij}$) if $\gamma_{ij} \geq \zeta$ (we will discuss about the similarity threshold, ζ in next section). We remove all the matrix elements of i 'th row and j 'th column from M . We repeat the finding of the maximum-valued matrix-element, γ_{ij} adding it to ρ and removing all the matrix elements of the corresponding row and column until either $\gamma_{ij} < \zeta$, or $m - \delta - |\rho| = 0$, or both.

Step 6: We sum up all the elements in ρ and add δ to it to get a total score. We multiply this total score by the reciprocal harmonic mean of m and n to obtain a balance similarity score between 0 and 1, inclusively.

$$\text{Similarity Score}(R_i, S_j) = \frac{(\delta + \sum_{i=1}^{|\rho|} \rho_i) \times (m + n)}{2mn}$$

Algorithm $MCLCS_1$

Input: r_i, s_j // r_i and s_j are two input strings where $|r_i| = \tau, |s_j| = \eta$
// and $\tau \leq \eta$ as mentioned earlier.

1. $\tau \leftarrow |r_i|, \eta \leftarrow |s_j|$
2. **while** $|r_i| \geq \zeta$ // we usually set ζ to 1. Details are discussed in next section.
3. **if** $r_i \in S_j$ // i.e., $S_j \cap r_i = r_i$
4. **return** r_i
5. **else** $r_i \leftarrow r_i \setminus c_\tau$ // i.e., remove the right-most character from r_i
6. **end if**
7. **end while**

Output: r_i // r_i is the Maximal Consecutive LCS starting at character 1

Figure 4.2: Maximal Consecutive LCS starting at character 1

Algorithm $MCLCS_n$

Input: r_i, s_j // r_i and s_j are two input strings where $|r_i| = \tau$, $|s_j| = \eta$
// and $\tau \leq \eta$ as mentioned earlier.

1. **while** $|r_i| \geq \zeta$ // we usually set ζ to 1. Details are discussed in next section.
2. determine all n -gram from r_i where $n = |r_i|$ and \bar{r}_i is the set of n -gram
3. **if** $x \in s_j$ where $\{x \mid x \in \bar{r}_i, x = \mathbf{Max}(\bar{r}_i)\}$ // i is the number of n -grams
4. // $\mathbf{Max}(\bar{r}_i)$ returns the maximum length n -gram from \bar{r}_i
5. **return** x
6. **else** $\bar{r}_i \leftarrow \bar{r}_i \setminus x$ // remove element x from set \bar{r}_i
7. **end if**
8. **end while**

Output: x // x is the Maximal Consecutive LCS starting at any character n

Figure 4.3: Maximal consecutive LCS starting at any character n

Algorithm $similarityMatching$

Input: r_i, s_j // r_i and s_j are two input words where $|r_i| = \tau$, $|s_j| = \eta$ and $\tau \leq \eta$.

1. $v \leftarrow \mathbf{SOCPMI}(r_i, s_j)$ // This method determines semantic similarity
2. // between two words. Any other similarity
3. // method can also be used instead.
4. **if** $v > \lambda$ // We discuss about λ in next section.
5. $v \leftarrow 1$
6. **else** $v \leftarrow v / \lambda$
7. **end if**

Output: v // v is the semantic similarity value between 0 and 1, inclusively

Figure 4.4: Similarity matching

4.3.1 Choosing the values of ζ , λ and ς

ζ is the minimum number of characters for which we continue the matching process. Theoretically ζ could be any value between 1 and m inclusively. We usually set ζ to 1. If we use ζ to 1 then we can get expected matching result for small-length tokens. e.g., if we have three sample tokens named *min*, *max* and *similarity* and we set ζ to 1. The pair *min max* returns m and the pair *min similarity* returns \emptyset when we use $MCLCS_1$. When we use $MCLCS_n$, the first pair returns m and the second pair returns mi . But if we set ζ to 2, the pair *min max* returns \emptyset for both $MCLCS_1$ and $MCLCS_n$. If we set ζ to 3, the pair *min similarity* returns \emptyset for both $MCLCS_1$ and $MCLCS_n$. Basically, λ is dependent on the semantic similarity method we use. We choose the value of λ based on the maximum range of the similarity values for the semantic similarity method we use. We usually set λ to 20 when we use *SOCPMI* semantic similarity method because our experiments showed that 20 is the maximum of the region of values for best matches. We can use any other similarity measure including a dictionary-based or a hybrid approach. E.g., if we use the Roget-based measure (Jarmasz and Szpakowicz, 2001) than we need to set λ to 16. One of the main advantages of using distributional measures based on corpus is that it covers significantly more tokens than any dictionary-based measure. Theoretically, ς could be any value between 0 and 1 exclusively, but we usually set ς close to 0 (we set $\varsigma = 0.01$ for all of our experiments). All matrix elements having values lower than ς may have negative impacts to the matching result, thus it is better to omit those.

4.4 Walk-Through Examples

We provide two examples that describe the proposed method and determine the similarity score. In example 1, we use two real element names from a database schema and in example 2, we use two element names that we created, in order to better illustrate the method (to cover all its strength at once).

4.4.1 Example 1

Let $R_i = \text{"maxprice"}$, $S_j = \text{"High_Price"}$.

Step 1: After eliminating all special characters and punctuations, if any, and then using word segmentation method and lemmatizing , we get $R_i = \{max, price\}$ and $S_j = \{high, price\}$ where $m = 2$ and $n = 2$.

Step 2: Because only one token (i.e., *price*) in R_i exactly matches with S_j we set δ to 1. We remove *price* from both R_i and S_j . So, $R_i = \{max\}$ and $S_j = \{high\}$. As $m - \delta \neq 0$, we proceed to next step.

Step 3: We construct a 1×1 *matching matrix*, M_1 . Consider the *max high* pair where $\eta = 4$ is the length of the larger token (*high*), $\tau = 3$ is the length of the smaller token (*max*) and 0 is the maximal length of the consecutive portions of the smaller token that consecutively match with the larger token. So, $v_1 = v_2 = v_3 = 0$ and $\alpha_{11} = 0$.

$$M_1 = \begin{matrix} & \begin{matrix} high \\ \end{matrix} \\ \begin{matrix} max \\ \end{matrix} & \begin{bmatrix} 0 \end{bmatrix} \end{matrix}$$

Step 4: We construct a 1×1 *similarity matrix*, M_2 . Here, $\lambda = 20$ as we used the *SOCPMI* method.

$$M_2 = \begin{matrix} & \begin{matrix} high \\ \end{matrix} \\ \begin{matrix} max \\ \end{matrix} & \begin{bmatrix} 0.326 \end{bmatrix} \end{matrix}$$

Step 5: We construct a 1×1 *joint matrix*, M and assign equal weight factor by setting both ψ and ϕ to 0.5.

$$M = \begin{matrix} & \begin{matrix} high \\ \end{matrix} \\ \begin{matrix} max \\ \end{matrix} & \begin{bmatrix} 0.163 \end{bmatrix} \end{matrix}$$

We find the only maximum-valued-matrix-element, $\gamma_{ij} = 0.163$ and add it to ρ as $\gamma_{ij} \geq \varsigma$ (we use $\varsigma = 0.01$ in this example). So, $\rho = \{0.163\}$. The new M is empty after removing i 'th ($i = 1$) row and j 'th ($j = 1$) column. We proceed to next step as $m - \delta - |\rho| = 0$. (Here, $m = 2$, $\delta = 1$ and $|\rho| = 1$)

Step 6:

$$\begin{aligned}
\text{Similarity Score}(R_i, S_j) &= \frac{(\delta + \sum_{i=1}^{|\rho|} \rho_i) \times (m + n)}{2mn} \\
&= (1 + 0.163) \times 4/8 \\
&= 0.582
\end{aligned}$$

4.4.2 Example 2

Let R_i = “allmileage_make_maxkm”, S_j = “make_minmile_distance_possible_take”.

Step 1: After eliminating all special characters and punctuations, if any, and then using word segmentation method and lemmatizing, we get $R_i = \{all, mileage, make, max, km\}$ and $S_j = \{make, min, mile, distance, possible, take\}$ where $m = 5$ and $n = 6$.

Step 2: Only one token (i.e., *make*) in R_i exactly matches with S_j therefore we set δ to 1. We remove *make* from both R_i and S_j . So, $R_i = \{all, mileage, max, km\}$ and $S_j = \{min, mile, distance, possible, take\}$. As $m - \delta \neq 0$, we proceed to next step.

Step 3: We construct a 4×5 matching matrix, M_1 . Consider the *mileage possible* pair where $\text{length}(LCS(\text{mileage}, \text{possible})) = 3$, $\eta = 8$ is the length of the larger token (*possible*), $\tau = 7$ is the length of the smaller token (*mileage*) and 2 is the maximal length of the consecutive portions of the smaller token that consecutively match with the larger token, where matching starts from 3rd character of the smaller token and 7th character of the larger token. So, $v_1 = 3^2 / (8 \times 7) = 0.16$

$$v_2 = 0$$

$$v_3 = 2^2 / (8 \times 7) = 0.071$$

$$\text{and } \alpha_{24} = 0.33 \times v_1 + 0.33 \times v_2 + 0.33 \times v_3$$

$$= 0.076$$

$$M_1 = \begin{matrix} & & \begin{matrix} min & mile & distance & possible & take \end{matrix} \\ \begin{matrix} all \\ mileage \\ max \\ km \end{matrix} & \left[\begin{array}{ccccc} 0 & 0.055 & 0.041 & 0.027 & 0.082 \\ 0.188 & 0.565 & 0.058 & 0.076 & 0.058 \\ 0.11 & 0.082 & 0.027 & 0 & 0.055 \\ 0.11 & 0.082 & 0 & 0 & 0.123 \end{array} \right] \end{matrix}$$

Step 4: We construct a 4×5 similarity matrix, M_2 . Here, $\lambda = 20$ as we used SOCPMI method.

$$M_2 = \begin{matrix} & & \begin{matrix} min & mile & distance & possible & take \end{matrix} \\ \begin{matrix} all \\ mileage \\ max \\ km \end{matrix} & \begin{bmatrix} 0.172 & 0.233 & 0.48 & 0 & 0.813 \\ 0.587 & 0.976 & 0.826 & 0 & 0.558 \\ 0.199 & 0.194 & 0.141 & 0 & 0.243 \\ 0.67 & 0.962 & 0.89 & 0 & 0.408 \end{bmatrix} \end{matrix}$$

Step 5: We construct a 4×5 joint matrix, M and assign equal weight factor by setting both ψ and ϕ to 0.5.

$$M = \begin{matrix} & & \begin{matrix} min & mile & distance & possible & take \end{matrix} \\ \begin{matrix} all \\ mileage \\ max \\ km \end{matrix} & \begin{bmatrix} 0.086 & 0.144 & 0.26 & 0.013 & 0.447 \\ 0.388 & 0.771 & 0.442 & 0.038 & 0.308 \\ 0.154 & 0.138 & 0.084 & 0 & 0.149 \\ 0.39 & 0.522 & 0.445 & 0 & 0.266 \end{bmatrix} \end{matrix}$$

We find the maximum-valued matrix-element, $\gamma_{ij} = 0.771$ and add it to ρ as $\gamma_{ij} \geq \varsigma$ (we use $\varsigma = 0.01$ in this example). So, $\rho = \{0.771\}$. The new M after removing i 'th ($i = 2$) row and j 'th ($j = 2$) column is:

$$M = \begin{matrix} & & \begin{matrix} min & distance & possible & take \end{matrix} \\ \begin{matrix} all \\ max \\ km \end{matrix} & \begin{bmatrix} 0.086 & 0.26 & 0.013 & 0.447 \\ 0.154 & 0.084 & 0 & 0.149 \\ 0.39 & 0.445 & 0 & 0.266 \end{bmatrix} \end{matrix}$$

We find the maximum-valued matrix-element, $\gamma_{ij} = 0.447$ for this new M and add it to ρ as $\gamma_{ij} \geq \varsigma$. So, $\rho = \{0.771, 0.447\}$. The new M after removing i 'th ($i = 1$) row and j 'th ($j = 4$) column is:

$$M = \begin{matrix} & \begin{matrix} \text{min} & \text{distance} & \text{possible} \end{matrix} \\ \begin{matrix} \text{max} \\ \text{km} \end{matrix} & \begin{bmatrix} 0.154 & 0.084 & 0 \\ 0.39 & 0.445 & 0 \end{bmatrix} \end{matrix}$$

Here, 0.445 is the maximum-valued matrix-element and $\gamma_{ij} \geq \zeta$. So, $\rho = \{0.771, 0.447, 0.445\}$. The new M after removing i 'th ($i = 2$) row and j 'th ($j = 2$) column is:

$$M = \begin{matrix} & \begin{matrix} \text{min} & \text{possible} \end{matrix} \\ \text{max} & \begin{bmatrix} 0.154 & 0 \end{bmatrix} \end{matrix}$$

We find 0.154 as the maximum-valued matrix-element and $\gamma_{ij} \geq \zeta$. So, $\rho = \{0.771, 0.447, 0.445, 0.154\}$. The new M is empty after removing i 'th ($i = 1$) row and j 'th ($j = 1$) column.

We proceed to next step as $m - \delta - |\rho| = 0$. (Here, $m = 5$, $\delta = 1$ and $|\rho| = 4$)

Step 6:

$$\begin{aligned} \text{Similarity Score } (R_i, S_j) &= \frac{(\delta + \sum_{i=1}^{|\rho|} \rho_i) \times (m + n)}{2mn} \\ &= (1 + 1.817) \times 11/60 \\ &= 0.516 \end{aligned}$$

4.5 Evaluation and Experimental Results

We now present experimental results that demonstrate the performance of our method. All the schemas we used in our experiments are from Madhavan et al. (2005), where they used web form schemas from two different domains, *auto* and *real estate*. Web form schema matching is the problem of identifying corresponding input fields in the web forms. Each web form schema is a set of elements, one for each input. The properties of each input include: the hidden input name or element name that is passed to the server when the form is processed, the description text and sample values in the option box. We tested on the same data as Madhavan et al. (2005), all of it, while they used 75% of it, randomly selected. We could not reproduce the exact 75% that they used. Figure 4.5 and

Figure 4.6 are two sample schemas from *auto* domain (vname are the element names to be matched), while Figure 4.7 is their manual mapping (the tags <left> and <right> are used to show an element name from the first schema that matches with an element name from the second schema).

```

<domain name="Automobile">
<search>
<source name="401carfinder"
url="http://www.401carfinder.com/ad_search_form.htm" />
<element>
<element vname="Region" text="Region">
<selectbox>
  <item>Alberta</item>
  <item>British Columbia</item>
  <item>Nova Scotia</item>
  <item>Ontario: Windsor</item>
  <item>Quebec</item>
  <item>USA</item>
</selectbox>
</element>
<element vname="MAKE" text="Make">
<selectbox>
  <item>All Makes</item>
  <item>Acura</item>
  <item>BMW</item>
  <item>Cadillac</item>
  <item>Yamaha</item>
</selectbox>
</element>
<element vname="Model" text="Model (optional)">
  <textbox />
</element>
<element vname="MinYear" text="Year (optional)">
  <textbox />
</element>
<element vname="MaxYear" text="to">
  <textbox />
</element>
<element vname="MaxPrice" text="Max. Price (optional)">
  <textbox />
</element>
<element vname="Keyword" text="Keyword (optional)">
  <textbox />
</element>
<element vname="Sort" text="sort results by">
<RadioButton>
  <item>price</item>
  <item>year</item>
  <item>make</item>
</RadioButton>
</element>
</search>
</domain>

```

Figure 4.5: A sample schema named "401carfinder" from *auto* domain

```

<domain name="Automobile">
<search>
<source name="AutoWeb"
url="http://www.autoweb.com/content/buy/used/index.cfm" />
<element>
<element vname="vehicle" text="Select Vehicle">
<selectbox>
  <item>Acura CL</item>
  <item>Volvo 240</item>
  <item>Yugo Cabrio</item>
</selectbox>
</element>
<element vname="search_mileage_int" text="search within">
<selectbox>
  <item>25</item>
  <item>50</item>
  <item>100</item>
</selectbox>
</element>
<element vname="Entered_Postal_Code_vch" text="miles of">
  <textbox />
</element>
<element vname="Low_Price" text="Price Range">
<selectbox>
  <item>Any</item>
  <item>$5,000</item>
  <item>$15,000</item>
  <item>$20,000</item>
</selectbox>
</element>
<element vname="High_Price" text="to">
<selectbox>
  <item>Any</item>
  <item>$25,000</item>
  <item>$40,000 +</item>
</selectbox>
</element>
<element vname="numCarsOnOnePage" text="Vehicles per page">
<selectbox>
  <item>5</item>
  <item>10</item>
  <item>15 </item>
</selectbox>
</element>
</element>
</search>
</domain>

```

Figure 4.6: A sample schema named "AutoWeb" from *auto* domain

```
<mappings schema1="401carfinder" schema2="AutoWeb">
<match>
<left>
  <item>Region</item>
</left>
<right>
  <item>Entered_Postal_Code_vch</item>
</right>
</match>
<match>
<left>
  <item>MAKE</item>
</left>
<right>
  <item>vehicle</item>
</right>
</match>
<match>
<left>
  <item>Model</item>
</left>
<right>
  <item>vehicle</item>
</right>
</match>
<match>
<left>
  <item>MaxPrice</item>
</left>
<right>
  <item>High_Price</item>
</right>
</match>
</mappings>
```

Figure 4.7: Manual mapping between "401carfinder" and "AutoWeb"

In each domain, they manually created mappings between randomly chosen schema pairs. The matches were *one-many*, i.e., an element can match any number of elements in the other schema. These manually created mappings are used as a *gold standard* to compare the mapping performance of the different methods, including our method. Table 4.1 provides detailed information about each of the two domains and our results.

Domain name	Number of schemas	Number of manual mappings	Similarity threshold score in our method	Number of predicted mapping pairs	Number of correct mapping pairs	Number of manually created mapping pairs
Auto	30	95	0	33116	769	769
			0.1	961	628	
			0.2	769	596	
			0.3	701	564	
			0.4	689	558	
			0.5	642	530	
			0.6	501	424	
			0.7	438	382	
			0.8	200	192	
			0.9	176	176	
			1.0	103	103	
Real estate	20	57	0	4262	280	280
			0.1	364	232	
			0.2	310	211	
			0.3	248	176	
			0.4	228	173	
			0.5	203	164	
			0.6	155	130	
			0.7	124	105	
			0.8	59	55	
			0.9	48	48	
			1.0	30	30	

Table 4.1: Characteristics of the evaluation domains and our results

In each domain, we compared each predicted mapping pair against the manually created mapping pair. For our experiment, we only used element names for matching. We used eleven different similarity thresholds ranging from 0 to 1 with interval 0.1. E.g., using *auto* domain when we used similarity threshold 0.1, our method matched 961 elements, out of which 628 elements were among the 769 manually matched elements. Precision vs. similarity threshold curves and recall vs. similarity threshold curves of the two web domains for the eleven different similarity thresholds are shown in Figure 4.8 and Figure 4.9, respectively. P-R curves of the two web domains for the eleven different similarity thresholds are shown in Figure 4.10 where the similarity threshold decreases from left to right in the figure. Figure 4.11 shows F-measure vs. similarity threshold

curves; it is obvious that a lower similarity threshold (≈ 0.2) gives a better F-measure score.

The reason for a lower similarity threshold to obtain a better F-measure score is that we always take into accounts both the string similarity and semantic word similarity measures. If two strings have perfect semantic word similarity score (i.e. ≈ 1) and no string similarity score (i.e. ≈ 0), which is practically a perfect matching (e.g., *car* and *vehicle*); this lowers the total similarity score. Again, we multiply this total score by the reciprocal harmonic mean of m and n to obtain a balance similarity score which also lowers the final similarity value.

In Figure 4.9, when we use string similarity threshold score of 1 (i.e., matching the element names exactly, therefore no semantic similarity matching is needed), we obtain recall values of 0.133 and 0.107 for auto and real estate domains, respectively. We can consider these scores as the baselines.

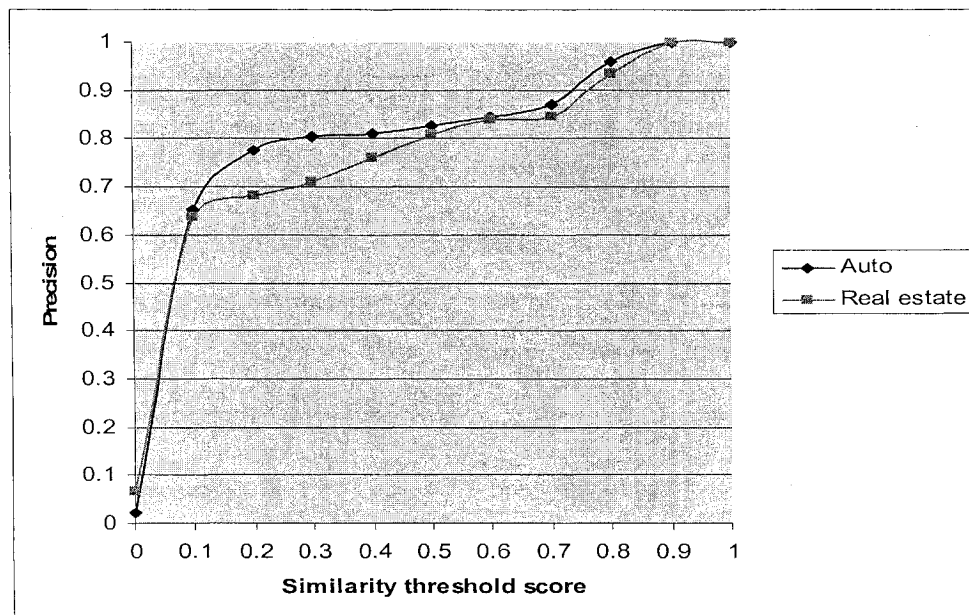


Figure 4.8: Precision vs. similarity threshold curves of the two web domains for eleven different similarity threshold

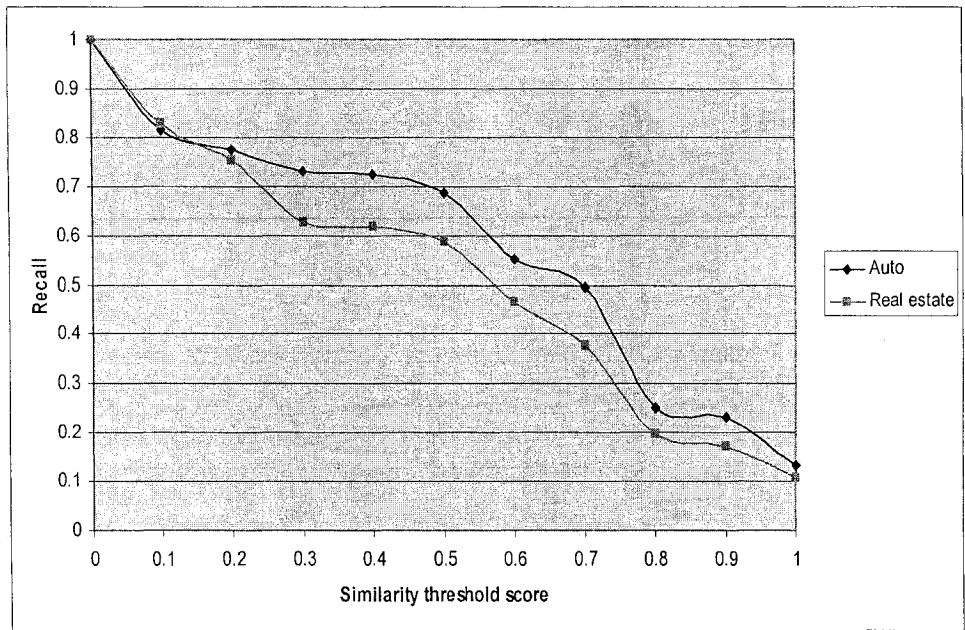


Figure 4.9: Recall vs. similarity threshold curves of the two web domains for eleven different similarity threshold

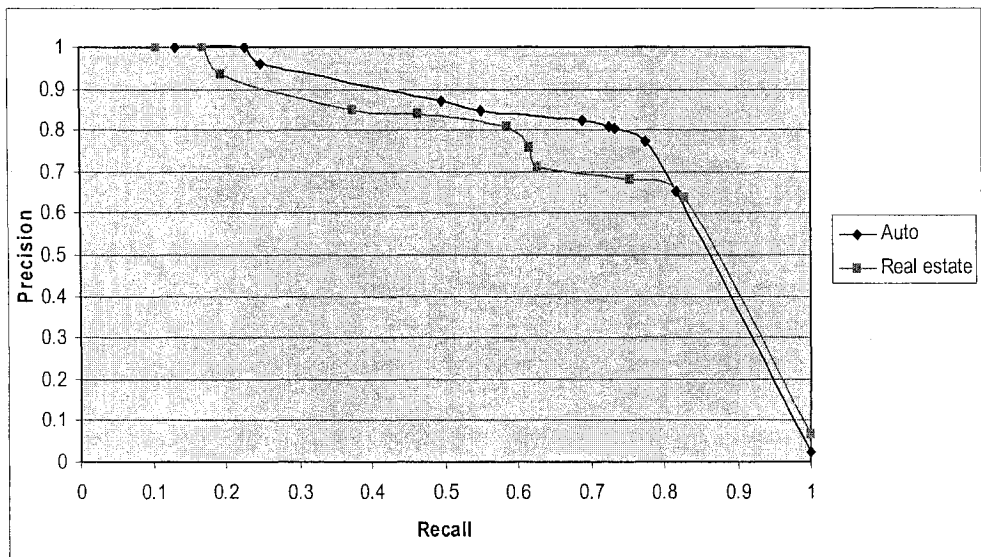


Figure 4.10: P-R curves of the two web domains for eleven different similarity threshold (similarity threshold decreases from left to right)

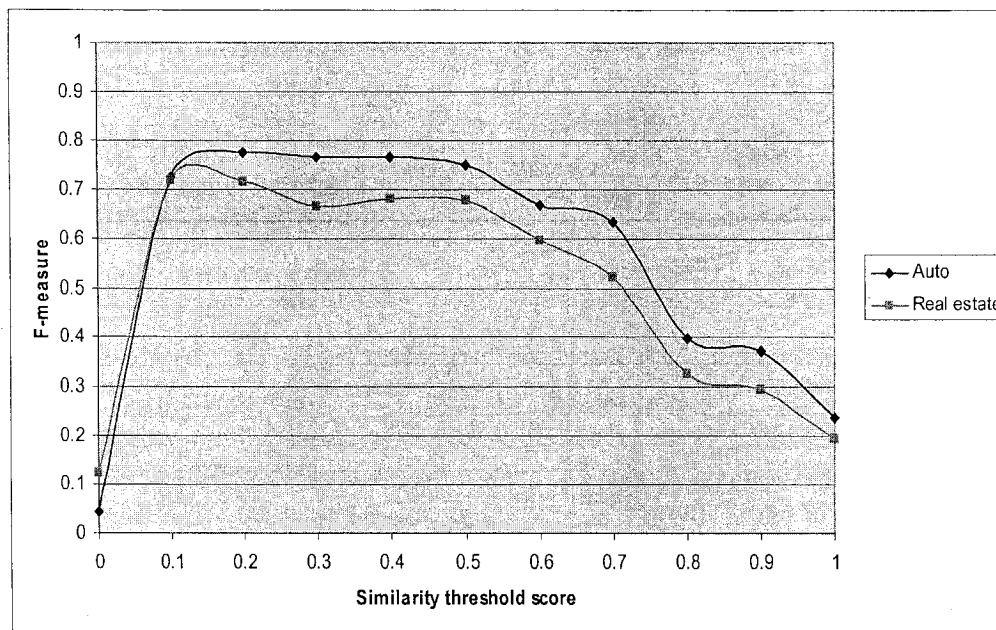


Figure 4.11: F-measure vs. similarity threshold curves of the two web domains for eleven different similarity threshold

Madhavan et al. (2005) used three methods: *direct*, *pivot* and *augment*. They selected a random 25% of the manually created mappings in each domain as training data and tested on the remaining 75% of the mappings. In the *augment* method, they used different base learners such as name learner, text learner, data instance learner, context learner and then used a meta-learner to combine the predictions of the different base learners into a single similarity score. To train a learner, the *augment* method requires learner specific positive and negative examples for the element on which it is being trained. The *direct* method uses the same base learners of *augment* method, but the training data for these learners is extracted only from the schemas being matched. *Pivot* is the method that computes cosine distance of the interpretation vectors of the two elements directly.

For *auto* domain, the *direct*, *pivot* and *augment* methods achieved precision of around 0.76, 0.74 and 0.92, recall of around 0.74, 0.78, 0.72 and F-measure of around 0.73, 0.74 and 0.78 respectively. We achieved around 0.78 as precision, recall and F-measure with 0.2 as similarity threshold.

For *real estate* domain, the *direct*, *pivot* and *augment* methods achieved precision of around 0.78, 0.71 and 0.76, recall of around 0.69, 0.74, 0.81 and F-measure of around 0.71, 0.71 and 0.78 respectively. We achieved precision of 0.68, recall of 0.75, and F-measure of 0.72 with 0.2 as similarity threshold.

Generally, it seems that precision does matter more than recall in the schema matching problem. But pragmatically it is not possible to determine fully automatically all matches between two schemas, and the implementation of the matching therefore only determine match candidates that are then verified by a human expert. If a human expert is involved in verification procedure then recall is as important as precision, i.e., F-measure does matter more than precision.

Our method is faster than the method of Madhavan et al. (2005) because it uses a single property in matching (element names), while their method uses multiple properties (names, descriptions, values). We could not quantify the amount of speed-up achieved by using a single property over multiple properties used by Madhavan et al. (2005) as they did not measure the time required to match.

4.6 Summary

In this chapter, we formulated a name-based schema matching method where we used word segmentation and corpus based semantic similarity. We also used a normalized and modified versions of LCS for string matching. We compared our results with the *augment* method of Madhavan et al., (2005) where in *auto* domain we achieved the same F-measure score, and in *real estate* domain we achieved 0.06 less F-measure score. One reason may be in *real estate* domain, the same elements can have very different names but they can have similarities in text description, context description and data instances that are not used by our method. Our method uses only the names of the elements and it perform similar with the more complex method of Madhavan et al. (2005). Therefore we believe that when adding the text description, context description and instance match, we can achieve very good results.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In this thesis, we addressed the task of database schema matching. First, we evaluated a new corpus-based word similarity measure, called SOC-PMI and compared it with existing word similarity measures. We performed intrinsic evaluation on the noun pairs mentioned earlier. We also performed a task-based evaluation: solving synonyms test questions. One of the main characteristics of the SOC-PMI method is that we can determine the semantic similarity of two words even though they do not co-occur within the window size at all in the corpus. Actually, we are considering the second order co-occurrences, as we are judging by the co-occurrences of the neighbor words, not only the co-occurrence of the two target words. This is not the case for PMI-IR and many other corpus-based semantic similarity measures.

Second, we proposed a word segmentation method that could be exploited in web search engine to provide better suggestion when the search text contains three or more words in the ‘desegmented’ part. The method can also effectively distill new words, special terms and proper nouns when the corpus covers a huge collection of both domain-dependent and domain-independent words, and it can effectively avoid statistical errors in

shorter strings which belong to a longer one. Experimental results show that our word segmentation method can segment words with high precision and high recall.

Finally, we exploit both of the semantic similarity and the word segmentation method in our proposed name-based element-level schema matching method.

Our schema matching method uses a single property (i.e., element name) for matching and achieves a comparable F-measure score with respect to the methods that use multiple properties (e.g., element name, text description, data instance, context description). If we use a single property instead of multiple properties, it can speed up the matching process which is important when schema matching is used in P2P data management or online query processing in P2P environment. Our method is scalable, in the sense that, if needed, we could also add other properties (i.e., text description and context description) to obtain a better schema matching result.

5.2 Future Work

- We plan to apply our proposed Second Order Co-occurrence PMI method to other tasks, such as measuring the semantic similarity of two texts and detecting semantic outliers in speech recognition transcripts. The SOCPMI method may also be helpful as a tool to aid in the automatic construction of the synonyms of a word. A very naïve approach would be as follows. First, we need to sort out the significant words list based on PMI values for the word (say, x) we are interested to find the synonyms. If there are n significant words in this words list, we will apply the SOC-PMI method for each possible pair mapping from x to n . Instead of taking the similarity value, we will consider all the second order co-occurrence types and sort out this types list based on PMI values. The words on the top of the list could be the best candidates for synonyms of the word.
- Our corpus-based word segmentation method can be extended as a hybrid method with some additions to the algorithms. The absence of type frequencies in dictionary means that we can only use the length of the types. In that case, we need to focus on what type to choose for the same length types that share

some common characters. Again, we can not choose whether we take the elements of *leftMaxMatching* or *rightMaxMatching* when both of them return the same number of elements as we can not use entropy rate for the absence of type frequencies. Future directions also involve integrating the current word segmentation algorithm into a larger system for comprehensive and context-based word analysis.

- Our proposed schema matching method together with the semantic similarity of words method can be further extended for the task of paraphrase recognition, entailment identification, and measuring the semantic similarity of texts. A corpus-based measure is useful to identify any similarity between words like *President* and *Clinton* from sentences ‘*Mr. President was supposed to visit Europe*’ and ‘*Mr. Clinton was supposed to visit Europe*’. The proposed schema matching method can also be updated in exactly the same way as the text similarity approach to exploit text description matching, another approach for schema matching.
- Incorporating equality of canonical name representations for special prefix/suffix symbols (e.g., CName → customer name, and EmpNO → employee number) will enhance the performance of the method. We tried with the Opau¹¹, a collection of lists of acronyms, abbreviations, and initialisms on the Word Wide Web, which has 353,494 entries from 128 different categories, but it even made the result worse. The reason is that misleading acronyms, abbreviations, or initialisms return lower string similarity and semantic word similarity scores.
- Our name-based schema matching method can be combined with rule-based methods to improve the accuracy in domain specific schema matching.

¹¹ <http://www.abbreviationz.com/>

Bibliography

- Allison, L., Dix, T. I. 1986. A Bit-String Longest-Common-Subsequence Algorithm. In *Information Processing Letters*, 23: 305-310.
- Batini, C., Lenzerini, M. and Navathe, S. B. 1986. A comparative analysis of methodologies for database schema integration. *ACM Computing Surveys*, 18(4):323–364.
- Bell, G.S. and Sethi, A. 2001. Matching records in a national medical patient index. In *Communications of the ACM (CACM)*, 44(9):83-88.
- Brent, M. 1999. An efficient, probabilistically sound algorithm for segmentation and word discovery. *Machine Learning*, 34:71–106.
- Brent, M. and Cartwright, T. 1996. Distributional regularity and phonotactics are useful for segmentation. *Cognition*, 61:93-125.
- Bright, M. W., Hurson, A.R. and Pakzad, S.H. 1994. Automated resolution of semantic heterogeneity in multi databases. In *Transactions on Database Systems (TODS)*, 19(2):212–253.

- Brill, E. 1994. Some advances in transformation-based part of speech tagging. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*. pp. 748–753, AAI Press/MIT Press.
- Brown, P. F., DeSouza, P. V., Mercer, R. L., Watson, T. J., Della Pietra, V. J. and Lai, J. C. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18:467-479.
- Budanitsky, Alexander and Hirst, Graeme. 2006. Evaluating WordNet-based measures of semantic distance. *Computational Linguistics*, 32(1).
- Buckley, C., Salton, J. A. and Singhal, A. 1995. Automatic query expansion using Smart: TREC 3. In *Proceedings of the third Text Retrieval Conference*, Gaithersburg, MD.
- Christiansen, M. and Allen, J. 1997. Coping with Variation in Speech Segmentation. In *Proceedings of GALA 1997: Language Acquisition: Knowledge Representation and Processing*, pp. 327-332.
- Christiansen, M., Allen, J. and Seidenberg, M. 1998. Learning to Segment Speech Using Multiple Cues: A Connectionist Model. *Language and Cognitive Processes*, 13:221-268.
- Church, K.W. and Hanks, P. 1990. Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16(1):22-29.
- Daelamans, W., Bosch, A. van den and Weijters, A. 1997. IGTrees: Using trees for compression and classification in lazy learning algorithms. *Artificial Intelligence Review*, 11:407–423.
- Dagan, I., Lee, L. and Pereira, F.C.N. 1999. Similarity based models of word cooccurrence probabilities. *Machine Learning*, 34(1-3):43-69.
- Dale, R., Moisl, H. and Somers, H. 2000. *Handbook of Natural Language Processing*. Marcel Dekker, Inc. New York.

- Deligne, S. and Bimbot, F. 1995. Language Modeling by Variable Length Sequences: Theoretical Formulation and Evaluation of Multigrams. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP-95)*.
- de Marcken, C. 1995. The Unsupervised Acquisition of a Lexicon from Continuous Speech. *Technical Report AI Memo No. 1558*, M.I.T., Cambridge, MA.
- Do, H. H. and Rahm, E. 2002. COMA – A System for Flexible Combination of Schema Matching Approaches. In *Proceedings of the 28th International conference on Very Large Data Bases (VLDB)*, pp. 610-621.
- Dunning, T. 1993. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 19:61-74.
- Fung, P. and Wu D. 1994. Improving Chinese tokenization with linguistic filters on statistical lexical acquisition. *Fourth Conference Applied Natural Language Processing*, Stuttgart, pp. 180-181.
- Gao, J., Li, M., Wu, A. and Huang, Chang-Ning. 2005. Chinese word segmentation and named entity recognition: a pragmatic approach. *Computational Linguistics*, 31(4).
- Grefenstette, G. 1993. Automatic thesaurus generation from raw text using knowledge-poor techniques. In *Making sense of Words, 9th Annual Conference of the UW Centre for the New OED and Text Research*.
- Grefenstette, G. 1992. Finding Semantic Similarity in Raw Text: The Deese Antonyms. In: R. Goldman, P. Norvig, E. Charniak and B. Gale (eds.), *Working Notes of the AAAI Fall Symposium on Probabilistic Approaches to Natural Language*. AAAI Press, pp. 61-65.
- Hua, Y. 2000. Unsupervised word induction using MDL criterion. In *Proceedings ISCSL 2000*, Beijing.

- Inkpen, Diana and Désilets, Alain. 2005. Semantic Similarity for Detecting Recognition Errors in Automatic Speech Transcripts. In *Proceedings of Empirical Methods in Natural Language Processing (EMNLP 2005)*, Vancouver, Canada.
- Jarmasz, M. and Szpakowicz, S. 2003. Roget's thesaurus and semantic similarity, In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP-2003)*, Borovets, Bulgaria, pp. 212-219.
- Kang, Jaewoo and Naughton, J. F. 2003. On Schema Matching with Opaque Column Names and Data Values. In *Proceedings of Special Interest Group on Management Of Data (SIGMOD 2003)*, San Diego, CA
- Kit, C. and Wilks, Y. 1999. Unsupervised Learning of Word Boundary with Description Length Gain. In *Proceedings CoNLL99 ACL Workshop*. Bergen.
- Kondrak, G. 2005. N-gram similarity and distance. In *Proceedings of the Twelfth International Conference on String Processing and Information Retrieval (SPIRE 2005)*, Buenos Aires, Argentina, pp. 115-126.
- Landauer, T. K. and Dumais, S. T. 1997. A Solution to Plato's Problem: The Latent Semantic Analysis Theory of the Acquisition, Induction, and Representation of Knowledge. *Psychological Review*, 104(2):211-240.
- Lee, L. 1999. Measures of distributional similarity. In *Proceedings of the Association for Computational Linguistics (ACL-1999)*, pp. 23-32.
- Lesk, M. E. 1969. Word-word associations in document retrieval systems. *American Documentation*, 20(1):27-38.
- Lesk, M. E. 1986. Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone. In *Proceedings of the conference of Special Interest Group on Design Of Communication (SIGDOC)*, Toronto.
- Li, H. and Abe, N. 1998. Word clustering and disambiguation based on co-occurrence data. In *Proceedings of the joint conference of the International Committee on*

Computational Linguistics and the Association for Computational Linguistics (COLING-ACL), pp. 749-755.

- Lin, C. Y. and Hovy, E. H. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of Human Language Technology Conference (HLT-NAACL 2003)*, Edmonton, Canada.
- Lin, D. 1998. Automatic retrieval and clustering of similar words. In *Proceedings of the joint conference of the International Committee on Computational Linguistics and the Association for Computational Linguistics (COLING-ACL)*, pp. 768-774.
- Lin, D. 1998a. An information-theoretic definition of similarity. In *Proceedings of the 15th International Conference on Machine Learning*.
- Madhavan, J., Bernstein, P., Doan, A. and Halevy, A. 2005. Corpus-based Schema Matching. In *International Conference on Data Engineering (ICDE-05)*.
- Melamed, I. D. 1999. Bitext maps and alignment via pattern recognition. *Computational Linguistics*, 25(1), 107–130.
- Mikheev, A. 1997. Automatic rule induction for unknown word guessing. *Computational Linguistics*, 23(3), 405–423.
- Miller, G. A. and Charles, W. G. 1991. Contextual correlates of semantic similarity. *Language and Cognitive Processes*, 6(1): 1-28.
- Miller, G. A. 1995. WordNet: A Lexical Database for English, *Communications of the ACM*, 38 (11):39-41.
- Milo, T and Zohar, S. 1998. Using Schema Matching to Simplify Heterogeneous Data Translation. In *Proceedings of the International conference on Very Large Data Bases (VLDB)*, pp.122-133.
- Pantel, P. and Lin, D. 2002. Discovering word senses from text. In *Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 613-619.

- Pedersen, T., Patwardhan, S. and Michelizzi, J. 2004. WordNet::Similarity - Measuring the Relatedness of Concepts, in *Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI-04)*, July 25-29, San Jose, CA (Intelligent Systems Demonstration).
- Peng, F. and Schuurmans, D. 2001. A Hierarchical EM Approach to Word Segmentation, In *Proceedings of the Sixth Natural Language Processing Pacific Rim Symposium (NLPRS 2001)*. pp 475-480, Tokyo, Japan.
- Rabiner, L. 1989. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. In *Proceedings of IEEE*, 77(2).
- Rahm, Erhard and Bernstein, Philip A. 2001. A survey of approaches to automatic schema matching. *The International Journal on Very Large Data Bases (VLDB)*, 10(4):334-350.
- Rao, C. R. 1983. Diversity: Its measurement, decomposition, apportionment and analysis. *Sankhya: Indian Journal of Statistics*, 44(A):1-22.
- Resnik, P. 1995. Using information content to evaluate semantic similarity. In *Proceedings of 14th International Joint Conference on Artificial Intelligence*, Montreal, pp. 448-453.
- Rosenfeld, R. 1996. A maximum entropy approach to adaptive statistical language modeling. *Computer speech and language*. 10:187-228.
- Rubenstein, H. and Goodenough, J. B. 1965. Contextual correlates of synonymy. *Communications of the ACM*, 8(10): 627-633.
- Rumelhart, D. E. and McClelland, J. 1986. On learning the past tense of English verbs. In *Parallel distributed processing* Vol. II, pp. 216–271. Cambridge, MA: MIT Press.
- Saffran, J. R., Newport, E. L. and Aslin, R. N. 1996. Word segmentation: The role of distributional cues. *Journal of Memory and Language*, 35:606–621.

- Salton, G. and McGill, M.J. 1983. Introduction to Modern Information Retrieval. *McGraw-Hill*.
- Seligman, L., Rosenthal, A., Lehner, P., and Smith, A. 2002. Data integration: Where does the time go? *Bulletin of the Technical Committee on Data Engineering*, 25(3).
- Shannon, C. E. and Weaver, W. 1963. *The mathematical theory of communication*. Urbana: University of Illinois Press.
- Sproat, R., Shih, C., Gale, W. and Chang, N. 1996. A stochastic finite-state word-segmentation algorithm for Chinese. *Computational Linguistics*, 22(3):377-404.
- Sproat, R., Shih, C., Gale, W. and Chang, N. 1994. A stochastic word segmentation algorithm for a Mandarin text-to-speech system. *32nd Annual Meeting of the Association for Computational Linguistics*, Las Cruces, NM, pp 66-72.
- Turney, P. D. 2001. Mining the Web for Synonyms: PMI-IR versus LSA on TOEFL. In *Proceedings of the Twelfth European Conference on Machine Learning (ECML-2001)*, pp. 491-502.
- Vechtomova, O. and Robertson, S. 2000. Integration of collocation statistics into the probabilistic retrieval model. In *22nd Annual Colloquium on Information Retrieval Research*, Cambridge, England.
- Weeds, J., Weir, D. and McCarthy, D. 2004. Characterising measures of lexical distributional similarity. In *Proceedings of the 20th International Conference on Computational Linguistics, COLING-2004*. Geneva, Switzerland. pp. 1015-1021
- Xu, J. and Croft, B. 2000. Improving the effectiveness of information retrieval. *ACM Transactions on Information Systems*, 18(1):79-112.
- Yarowsky, D. 1992. Word-sense disambiguation using statistical models of Roget's categories trained on large corpora. In *Proceedings of the International Conference on Computational Linguistics (COLING-92)*, Nantes, France, pp. 454-460.