Intelligent Typo Correction and Text Categorization Using Machine Learning and Ontology Networks

by

Yinghao Huang

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy (Information Systems Engineering) in the University of Michigan-Dearborn 2014

DEDICATION

I dedicate this dissertation to my beloved family – to my parents, as without their love and support over the years this work would not be possible, and to my loving wife, Yi Gao, and my dear daughter, Mackenzie, who make our life an inspirational journey.

ACKNOWLEDGEMENTS

I would like to thank my advisor Dr. Yi Lu Murphey for her valuable guidance, advices, support, and the inspirational discussions we had. I thank her for not only training me with professional research abilities, but also teach me rules and principles that will become knowledge and wisdom in my future. I would also like to thank members of my doctoral committee, Professors Bill Grosky, Paul Watta and Hafiz Malik for their suggestions and guidance.

I have been lucky to be a part of a group of enthusiastic researchers at the UM – Dearborn ISL lab. I appreciate the opportunities for collaborations and discussions with variety of people, including Jungme Park, Chen Fang, Qi Dai, Wenduo Wang, Shen Xu, Dai Li, Xipeng Wang, etc. It has been a precious experience to work with them and learn from them. Special thanks to Xipeng Wang, for helping me with experiments and having discussion with me about research problems and exchange ideas.

Finally, I would like to thank my wife, Yi Gao, for her support over the years and encouragement for continuing my research work.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

In this dissertation, we present our research work in text mining field, mainly focusing on accurately and efficiently performing the task of text preprocessing and text categorization, using machine learning techniques and ontology networks. Specifically, an innovative intelligent typo correction system, ITDC, is proposed to automatically correct misspellings in text documents using general language knowledge and domain specific knowledge extracted by machine learning algorithms. It has the capability of correcting a broad range of typos, from simple typos such as duplication, omission, transposition, substitution characters, to complex spelling errors, such as word boundary errors, unconventional use of acronyms, and multiple versions of abbreviations of the same words. It uses the generated knowledge for identifying unconventional acronyms, grouping similar words (correctly spelled and misspelled), and ranking correction candidates.

An innovative text categorization model, VSM_WN_TM, is also presented. VSM_WN_TM is a special Vector Space Model (VSM) that incorporates word frequencies, ontology networks and latent semantic information. Unlike the traditional text representation using only Bag-of-words (BOW) features, it also incorporates semantic and syntactic relationship among words such as synonymy, co-occurrence and context, with the purpose of providing more inclusive and accurate text representation.

The results from the performed experiments are highly encouraging. The ITDC system is evaluated through a case study that involves the automatic processing of automotive fault diagnostic text documents. The performance generated from more than 580000 automotive fault diagnostic documents provided by two different automotive manufacturers show that the proposed system outperforms state-of-art spell checking systems.

The proposed VSM_WN_TM model is evaluated on three publicly available datasets and one domain-specific dataset. Experiment results show that our approach significantly improves text classification by outperforming baseline approaches such as using only latent features and traditional VSM approaches.

Indexed Keywords: text mining, text preprocessing, text categorization, typo correction, machine learning, ontology networks, statistical language modeling.

CHAPTER 1. INTRODUCTION

1.1. Motivation

We have witnessed the blooming of web-based text resource volume in recent decades within different domains, such as social networks (Email, Twitter, Micro blogs, customer reviews, etc.), automotive industries (vehicle diagnostics, customer queries, etc.) and medical science (physician diagnostics, patient medical records, etc.). With the proliferation of text data, text mining techniques have drawn significant attention by people in both academic and industrial field for quite a long time. Text mining is generally defined as finding interesting patterns and trends of data. Especially in the age of "Big Data" [1], when data management systems are overloaded daily by data in free text form, text document processing brings more and more necessity for automated, accurate and efficient text mining algorithms. These algorithms could be used to fulfill different objectives, such as data preprocessing, text clustering and categorization, information retrieval, etc.

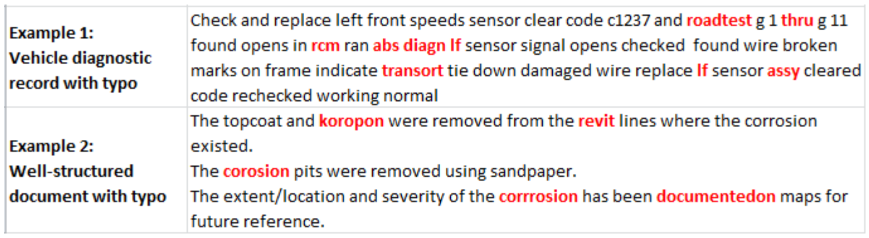
Text mining is characterized as the process of analyzing text to extract information that is useful for particular purposes. The key point that makes text mining so special in the field of data mining, which is already a well-developed area that is entering a mature phase [2], is that text needs to be transferred into numerical representations before conducting further analysis, and in most of the text mining applications, text has an unstructured and causally written form that causes difficulties for extracting useful information, such as clinical document analysis [3,4], emails, instant messages, automotive diagnostic text mining [5], etc. For example, a lot of spelling errors occur in text data and can be problematic for text data mining.

One of the most crucial tasks in the text mining field is text categorization. This is the task of building a classifier that assigns a pre-defined category to each text document in the text collection, the source of which highly depends on the problem and the application domain. Due to the fact that the document category is usually defined based on various application requirements, the typical approach of text categorization is to derive numerical features that represent text. After that, supervised machine learning techniques are used to train a classifier based on those extracted features and the categories of pre-classified text documents, to later perform categorization tasks on previously unseen text documents. As a result, the performance of text categorization task depends on the feature generation method used and the machine learning techniques adopted.

In this dissertation, we present our research work in intelligent text mining that focuses on accurately and efficiently performing the task in two main areas. The first area includes developing a system that automatically corrects misspellings in text documents so that they are much more comprehensible by machines and much more accurate for further processing such as text categorization. The second area is to develop a text categorization model that incorporates word based text features with semantic relationship learned from ontology networks and latent semantic structure information from statistical topic modeling. This hybrid text representation model could significantly help improving text categorization accuracy.

1.2. *Problem description and research focus*

As discussed in section 1.1, unstructured and free-style written text documents are commonly found in many text mining applications. These documents often do not follow grammar rules, and contain misspelled words, abbreviations and specific terminologies that are not found in standard English dictionaries and may be barely comprehensible to people outside the application field. We use an example to illustrate the different problem complexity comparing well-structured text documents with unstructured text documents. Fig. 1 illustrates those two document examples. In both documents, non-word misspellings are marked in red. It is obvious that unstructured documents are much harder to understand, and the typos are much more difficult to be identified and corrected. In terms of accuracy, misspelling correction in such applications is indisputably essential for automatic text retrieval, categorization or clustering systems, since typos can be misinterpreted in many different ways by automatic text processing systems. In terms of efficiency, automated intelligent typo correction is also of significant necessity in text mining applications that deals with vast amount of data, which makes manual inspection almost impossible.



Example 1: Vehicle diagnostic record with typo	Check and replace left front speeds sensor clear code c1237 and **roadtest** g 1 **thru** g 11 found opens in **rcm** ran **abs diagn lf** sensor signal opens checked found wire broken marks on frame indicate **transort** tie down damaged wire replace **lf** sensor **assy** cleared code rechecked working normal
Example 2: Well-structured document with typo	The topcoat and **koropon** were removed from the **revit** lines where the corrosion existed. The **corosion** pits were removed using sandpaper. The extent/location and severity of the **corrrosion** has been **documentedon** maps for future reference.

Fig. 1. Examples of well-structured and unstructured text documents

Many automatic spelling correction systems have already been developed to help people with their typing tasks, such as texting, web searching, etc. However, the current automatic typo correction technologies are still short in accuracy [6,7].

In text categorization, the document category is defined by users based on the application requirements, such as the topic that a news article discusses, the importance of a vehicle diagnostic record that describes vehicle repair details [5], and the fact stated in a medical diagnostic document that whether or not an injury condition sustains [4], etc. Consider the following automotive diagnostic records in two categories: The first record is defined as category A, which is an "important" document because it describes the "root cause" of the vehicle problem: "connector corroded". The second is defined as category B, which is unimportant, because it only describes the vehicle inspection process generally.

Category-A document: "perform abs self roadtest found rear wheel speeds sensor connector corroded into sensor replace sensor and connector road tester ok clear code"

Category-B document: "road roadtest traction control lamp on eec roadtest code c1280 u415 om rcm contact hot line 103912699 check connection at rcm check mounting bolts ok clear code"

To solve the above categorization problem, a typical approach of representing a text document is the Vector Space Model (VSM) [8], in which each document is represented by a weighted vector that provides a mathematical representation which is convenient for computation and analysis. However, this approach does not consider the semantic relationship between words, such as synonyms, hyponyms (IS-A relationship between words), etc. The classification accuracy is especially deteriorated in document sets where different categories have similar word occurrences [4, 9]. As a result, a text categorization model that captures underlying semantic and syntactic information besides single word occurrence features is of great necessity.

The major research focuses in this dissertation include:

- Automatic typo detection and correction for an unstructured and large-scale text corpus by incorporating general language knowledge and domain specific knowledge generated by machine learning algorithms.
- Incorporating ontology network information and machine learning techniques to automatic text categorization, by capturing semantic and syntactic relationship between words.

1.3. Major research contributions

The major original contributions of this dissertation include research work in intelligent typo correction and text categorization. First of all, we developed and implemented an intelligent typo detection and correction algorithm, which is a significant step forward towards fully-automatic spelling correction for processing large size corpora of unstructured text documents. Secondly, we propose a systematic way of building accurate text representations using single word information, as well as syntactic and semantic relationship between words to improve the performances of various text mining tasks, such as text categorization, text clustering, predictive analysis, information extraction, etc. Third, our approaches in these two fields can be combined together for other real-world applications that require text preprocessing and text categorization. Last but not least, they can be easily transplanted and applied to other text corpus, besides those discussed in this dissertation, e.g. text used in social networks such as instant messages and Twitter.

A summary of the major research achievements is presented here, including:

- Proposed an automated intelligent typo correction framework for unstructured and large-scale text collections, using general language knowledge and domain specific knowledge extracted by machine learning algorithms for identifying unconventional acronyms, grouping similar words (correctly spelled and misspelled), and ranking correction candidates.
- Proposed a hybrid text categorization approach that focuses on building VSM models with both ontology networks and statistical language model. We first generate the original

BOW representation by using an intuitive global weight scheme, and then build VSM models based on the WordNet ontology and statistical topic modeling.

- Evaluated our methods on publicly available datasets and domain-specific datasets:
  - Freeform technician verbatim problem descriptions (VDR)
  - Reuters-21578
  - Nist Topic Detection and Tracking corpus (TDT2)
  - 20 newsgroups
- Evaluated our system by comparing with state-of-art systems and baseline approaches:
  - Google spell checker
  - Aspell spell checker
  - Text categorization based on VSM only
  - Text categorization based on latent semantic feature only
- Analysis of text categorization performance, including the influence of number of latent topics, hyper weight of document connectivity in topic model, word class used for synonym set generation, and ontology weighting between words.
- Created open source packages for typo correction system, called ITDC, and text categorization system, called VSM_WN_TM, which can be adapted to other text mining applications.

1.4. Structure of dissertation

The remainder of this dissertation is organized as follows: Chapter 2 discusses the technical background and literature survey in the area of typo correction and text categorization. Chapter 3 introduces the ITDC typo correction system. Chapter 4 presents the VSM_WN_TM text categorization model and details of how to build text representation model using an ontology network and statistical language model. Chapter 5 discusses the details of our empirical case study, the evaluation of our system and the performance analysis. Finally, Chapter 6 gives an concise conclusion and discusses future work that can build on the ideas presented here.

CHAPTER 2. BACKGROUND AND RELATED WORK

This chapter describes the background information related to text mining, typo correction and ontology networks. Text representation models are also discussed including the vector space model and statistical language models. The chapter also gives a literature survey on relevant text categorization approaches including machine learning algorithms, statistical language models and ontology networks.

2.1 Text mining

Text mining, also known as knowledge discovery from text (KDT), is first mentioned in Feldman et al. [10]. It is defined as the applications of text processing and analysis that utilize combined techniques from information retrieval, natural language processing (NLP) that extract data from text, as well as machine learning and data mining algorithms, with the goal of finding useful patterns from text [11]. Decent estimates show that more than 80% of information is represented in the form of text [12], and this percentage will likely increase due to the continuous availability of online textual information. As a result, there has been significant development of text mining techniques in previous two decades.

A typical text mining tool first extracts a text document from text collection and conduct preprocessing steps, such as spell checking, removing special symbols or punctuations, tokenizing sentences into stream of words, etc. These procedures aim at providing a clean and understandable format of text for both human and machines. Even though there are plenty of efforts made to explore syntactic and semantic information from text, at the document level, most approaches are based on the concept that a text document is represented by a set of tokenized words; That is, a *bag-of-words* (BOW) [13]. The next step is to convert the cleaned text into numeric representations that are more appropriate for further automated processing and analysis by machines. The step is called *text encoding* [14], and state-of-art predominant approaches include vector space model [15] and statistical language models [16], which will be discussed in detail in section 2.4.

The *text analysis phase*, following with the text encoding, is used to extract interesting patterns or trends from text based on different application requirements. For instance, automatically label previously unseen documents based on user-defined category (text classification), find groups of documents with similar content (text clustering), extract parts of text and assign specific attributes (information extraction), etc. Common approaches here include machine learning, data mining and statistical analysis.

In summary, a typical work flow for text mining problems includes the following aspects:

- **Extracting information for human consumption,** including text summarization, document retrieval, information retrieval, etc.
- **Assessing document similarity**, including text categorization, document clustering, identifying key-phrases, etc.
- **Extracting structured information**, including entity extraction, information extraction, learning rules from text, etc.
- **Mining structured text**, including document clustering with links, wrapper induction, etc.

The scope of this dissertation, as mentioned in Chapter 1, mainly includes typo correction and text categorization based on machine learning, statistical language modeling techniques and external knowledge, which will also be discussed in detail in section 2.3 and 2.5.

2.2 Typo correction

As we mentioned earlier in Chapter 1, typo detection and correction is an important text process in many text mining applications. The major objective here is to reduce the errors made by machines due to the misinterpretation of misspelled text, and to provide a more understandable format for both human and machines for text mining tasks.

In this dissertation, our research focuses on correcting a broad range of typos, including simple duplication, omission, transposition, substitution characters, spelling errors, and unusual use of shorthand and acronyms. Typos can be divided into two major categories: “non-word errors” and “real-word errors”. Conventional spelling checkers typically use dictionaries to detect typos. Each term within a text document is compared against the valid words in a dictionary or a lexicon. Any term that does not match any word in the dictionary is flagged as an error. This kind of typos is called “non-word errors” [17,18], for instances, “abreviate” instead of “abbreviate”, “veruified” instead of “verified”, etc. Typing errors that result in a valid word, but not the one that the user intended, is called “real-word errors” [19,20], such as “font seat” instead of “front seat”, etc. The focus of our research in this paper is mainly on detecting and correcting non-word typos.

In general, two types of applications require typo corrections: the online programs involving text as input, and the offline automatic text document processing. Many online applications, such as web search engines and text based user interface programs, provide a list of correctly spelled words to user as he/she is typing each word on computers/phones/handheld devices.

Many automatic spelling correction systems have already been developed to help people with their typing tasks, such as texting, web searching, etc. However, the current automatic typo correction technologies are still short in accuracy [21,22]. In many text mining applications, such as text documents analysis, retrieval, and categorization, typos need to be detected and corrected automatically in order to achieve accurate results efficiently. Our solution is to combine domain specific knowledge with the general language knowledge to achieve accurate typo correction. In this research, we focus on the application of unstructured text mining in vehicle diagnostic records.

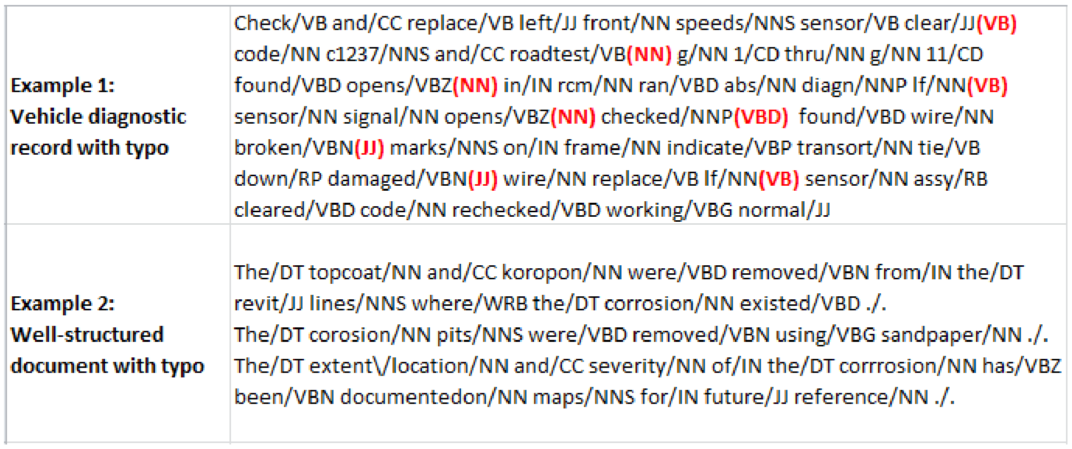
Discovering knowledge from unstructured text documents has many important applications including automotive fault diagnostics, medical document processing, and social network [23,24,25,26]. Large amounts of data have been collected in daily operations in many corporations, hospitals and government agencies, many of which are unstructured text data. The sheer volume of data makes manual or even semi-manual categorization or classification cumbersome and fallible. Automatic text categorization technologies have been developed and applied to many application problems including finding answers to similar questions or queries, classifying news by subject or newsgroup, categorizing web pages, organizing e-mail messages, etc. Many challenges exist in automatic text processing technologies, including representing semantics and abstract concepts and processing words with semantic ambiguity, such as polysemy and synonymy. Typos add another layer of complexity in automatic text document process.

The particular type of documents we are interested in has the following characteristics. These documents are short, and typically typed hastily in very short time period, e.g. in seconds or minutes, by people with varying education background and interests. Examples of such documents are short text messages, email text, medical descriptions of patients' symptoms, vehicle diagnostic documents such as customer's descriptions of vehicle problems, and technicians' descriptions of repair process, and etc. These documents often do not follow grammar rules, and contain misspelled words, abbreviations and domain specific terminologies that are not found in standard English dictionaries and barely comprehensible to people outside the application field.

Most of the current state-of-art typo correction systems are "interactive spelling checkers", which return multiple spell correction candidates, and allows user to select the intended correction [21]. Early in 1960s, researchers have already started working on text spelling error detection and correction. Many studies have been conducted with the purpose of developing correction techniques for non-word errors. Over time various approaches and successful systems were developed. Popular methods for finding misspellings and assessing suggestion candidates for misspellings include part-of-speech (POS) tagging [27,28], minimum editing distance [28,29], nearest neighbor search procedure, similarity key methods such as SoundEX systems and Metaphone algorithms [28,30], and modified version of Longest Common Subsequence algorithm [19]. Error correction methods, especially real-word error correction, are typically based on syntactic and semantic knowledge such as n-gram based techniques [17,32,33], and statistical learning from training data sets such as web documents are used as context to help correcting typos [33,34,20]. Many methods have been developed to create and rank correction candidate lists of detected typos, such as statistical language models [31], machine learning techniques [18,27,32,35], complex network approaches [36], and noisy channel models [33,37]. The spell checkers used for online applications mostly use online text as the resource for typo correction, such as news pages from the Web that are clean and well-spelled [33]; web queries input to search engines by Internet users [20]; and Google n-gram dataset [34,39,40]. These interactive spelling checkers discussed above require user to manually select a candidate to replace the typo, which are good for interactive software systems such as MS Office, but, not applicable to automatic document processing, such as document retrieval or classification, in which a system having the capability of performing fully automated error detection and correction is required.

A number of automatic typo correction algorithms have being developed so far. A typical approach is to select the best candidate generated for each typo based on word context or semantic information such as POS tagging [21,41,42]. Two interesting typo correction algorithms were proposed by Sebastian and David [41], one based on supervised learning and another unsupervised learning. The supervise learning algorithm uses a reverse edit distance method to generate a candidate list and then one on the list that has the highest score based on word occurrence or word's bigram stemmer score. The unsupervised algorithm attempts to correct spelling errors by first using low-frequency word as candidate typo, with the assumption that low-frequency word is usually a typo and a word will be misspelled in exactly the same way very few times. After the candidate typos are selected, the algorithm uses other words in the document set as valid lexicon, and select best candidate based on predefined rule set of word context. They reported over 90% accuracy generated by the supervised learning system on the 2200 misspellings words provided from a NASA database, and over 70% accuracy by the unsupervised system on 5833 misspellings from Orbiter Structure database. An example of text documents in the NASA database is shown in Fig. 1. However, for processing unstructured text documents, several issues need to be addressed. First, low-frequency words could also be valid words, especially those words in document sets that have smaller sizes. For example, in the vehicle diagnostic corpus we processed has more than 10% of the valid words only appear 1 time, such as “contaminate”, “erratic”, “unsecured” and so forth. Secondly, a word could be misspelled in the same way repeatedly due to the typing pattern of users. Last but not least, a single English dictionary as external source for typo detection and correction is insufficient, especially for unstructured domain-specific text with valid words not in typical lexicon. Patrick et al. developed an automatic spell correction system that takes advantage of the entire context surrounding misspelling [42]. They also explore the use of the part-of-speech for selecting candidates. The system they proposed has 97-98% of accuracy on over 600 randomly selected medical documents. However, this approach is based on the assumption that POS tagging result is reliable.

Although the systems discussed above all reported high accuracy on typo correction, they were developed and evaluated on text documents that were well written in terms of grammar, structure, and sentence and word boundary. For the unstructured and causally written text documents, such as the engineering diagnostic records [43] we are dealing with, these methods do not work well. We use an example to illustrate the different problem complexity in these two types of documents. Fig. 1 illustrates two document examples: one is a well-structured text, and the other is unstructured text. In both documents, misspellings were marked in red. The parsing results generated by using Stanford POS tagger [38] on the two documents in Fig. 1 are shown in Fig. 2, in which incorrect POS tags were followed by true POS tags marked by red. Obviously, POS tagging is not reliable when it is applied to the unstructured document, i.e. the vehicle diagnostic records, considering the ambiguity of sentence boundary and poor quality of grammar in the document.



Example 1: Vehicle diagnostic record with typo	Check/VB and/CC replace/VB left/JJ front/NN speeds/NNS sensor/VB clear/JJ**(VB)** code/NN c1237/NNS and/CC roadtest/VB**(NN)** g/NN 1/CD thru/NN g/NN 11/CD found/VBD opens/VBZ**(NN)** in/IN rcm/NN ran/VBD abs/NN diagn/NNP lf/NN**(VB)** sensor/NN signal/NN opens/VBZ**(NN)** checked/NNP**(VBD)** found/VBD wire/NN broken/VBN**(JJ)** marks/NNS on/IN frame/NN indicate/VBP transort/NN tie/VB down/RP damaged/VBN**(JJ)** wire/NN replace/VB lf/NN**(VB)** sensor/NN assy/RB cleared/VBD code/NN rechecked/VBD working/VBG normal/JJ
Example 2: Well-structured document with typo	The/DT topcoat/NN and/CC koropon/NN were/VBD removed/VBN from/IN the/DT revit/JJ lines/NNS where/WRB the/DT corrosion/NN existed/VBD ./. The/DT corosion/NN pits/NNS were/VBD removed/VBN using/VBG sandpaper/NN ./. The/DT extent\/location/NN and/CC severity/NN of/IN the/DT corrrosion/NN has/VBZ been/VBN documentedon/NN maps/NNS for/IN future/JJ reference/NN ./.

Fig. 2. Results of POS tagging on well-structured and unstructured text documents

The above discussion leads to our work in typo correction for processing unstructured documents with a focus on typo correction techniques for three types of non-word typos: word boundary errors, self-invented abbreviations, and ambiguous acronyms. We attempt to fill in the gap between the interactive and fully-automatic spelling correction techniques for processing large size corpora of unstructured text documents. The corrected text documents can then be used for further text processing, such as text document categorization and text document retrieval.

2.3 Ontology networks

Concerning that although machines can do a lot of things under human directions, they do not understand human language. People are always trying to find a way that makes machine process languages in a more sophisticated manner, besides the simple BOW representation. The basic idea here is that, if every document is marked or enriched by some knowledge that captures syntactic or semantic information between words, machines are able to "understand" text better. In the field of computer science and text mining, ontology network is generally defined as "a formal, explicit specifications of shared conceptualizations of a domain of interest that are shared by a group of people" [44]. Therefore, it provides a solution to facilitate text understanding and automatic processing of textual resources. It is explained in detail as follows by Ding and Foo, in [45]:

> " *'Conceptualization' refers to an abstract model of phenomena in the world by having identified the relevant concepts of those phenomena. 'Explicit' means that the type of concepts used, and the constraints on their use are explicitly defined. 'Formal' refers to the fact that the ontology should be machine readable. 'Shared' reflects that ontology should capture consensual knowledge accepted by the communities.*"

More specifically, the most widely used type of ontologies in text mining applications is called "terminological ontologies", which are mainly specified by subtype-supertype (IS-A) relations and describe concepts by using concept labels or synonyms [44]. Examples of well-known terminological ontologies include WordNet [49], Semantic Wiki [50], etc. Fig. 3 shows an example of WordNet ontology instance that explicitly provides relationship between words such as synonym, hyponym, etc. Since our focus in this dissertation is based on terminological ontologies, this will be further discussed in detail in Chapter 4.

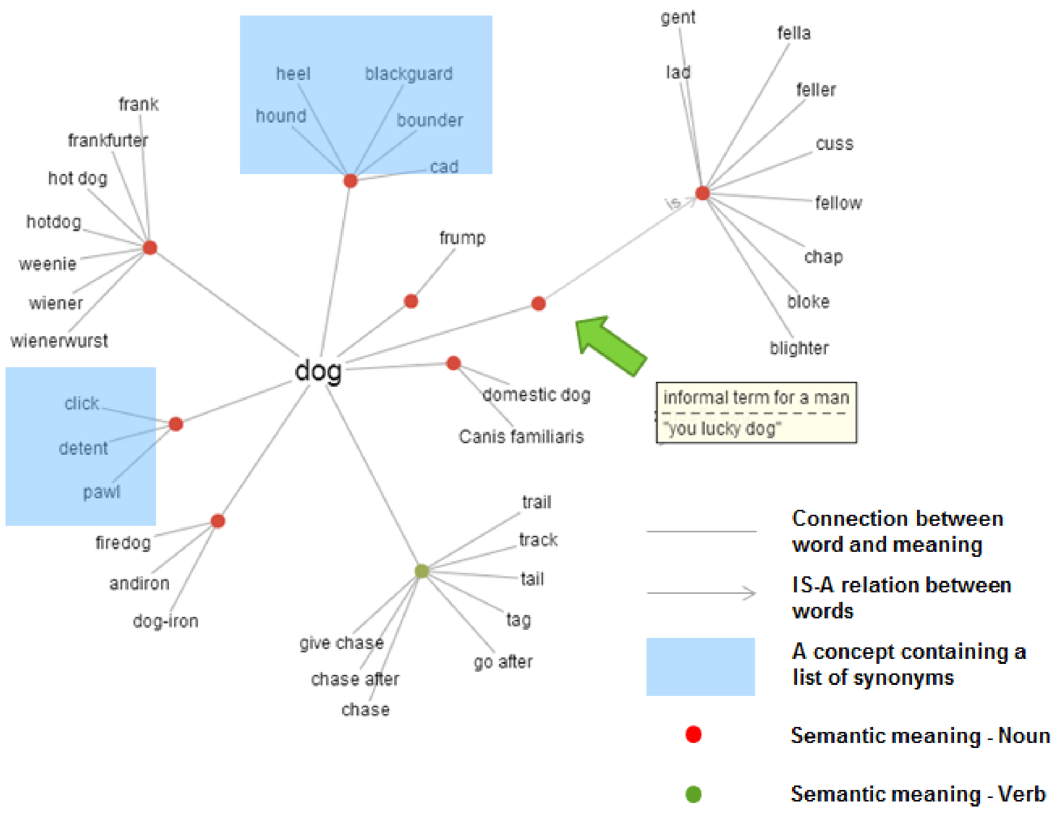


Fig. 3. Example of WordNet ontology instance visualization

In this research we define a concept as a set of synonyms that have similar semantic meanings in a specific application domain. In Fig. 3, concepts are highlighted in blue. The undirected connections connect word to its semantic meanings, and the directed connection represents IS-A relationship between words, which is also called as Hyponym/Hypernym relationship.

An immediate question for using ontology in text mining applications is that how to construct one that can be effectively used for data mining. Ontologies can be learnt from various resources such as structured, semi-structured or unstructured text corpus in specific domain, relational databases, publicly available taxonomies, etc. As a result, ontology learning techniques are divided into two groups: Constructing ontologies from scratch using unsupervised learning methods, and extending existent ontologies using supervised learning and classification methods [44,52,53]. The first approach usually requires a lot of manual or semi-manual work to build word ontology from scratch, and the adaptability of such ontology is usually restrained by domain-specific resources [50,54,55]. Therefore, to a large extent, current research work mainly focuses on learning ontologies from existing resources such as English lexicons [56]. Furthermore, most of the state-of-art approaches use only nouns for ontology building, and a large extent of methods aims at constructing IS-A-related concept hierarchies. [44,57,58].

Currently, a number of ontology learning algorithms has already been well-developed and powerful text ontology systems such as WordNet have generated and publicly available for use of academic or industry purpose. However, how to appropriately utilize such information effectively to facilitate various text data mining applications is still open for exploring. For example, in the field of text categorization, a mostly used approach is to use publicly available ontology to generate “Concept” level features as additional information to the text representation [46,47,48], while several issues still brings in great difficulties and challenges for solving the above problem, such as mapping ontology relationship to text feature representations, weighting word features based on ontology relationships, etc. [59,60].

As a result, this dissertation research focuses on how to incorporate information provided by ontology into text mining tasks such as text categorization, rather than on how to learn ontology from textual resources. We believe that a hybrid approach that combines conventional methods with appropriately generated ontology network information can enhance the quality of text representation, and thus improve the performances of text mining tasks using such representation. This leads to several well-known representation models developed by researchers for textual resources in recent decades, which will be discussed in the following section 2.4.

2.4 Text representation models

2.4.1 Vector Space Model (VSM)

As mentioned above in section 1.2, the conventional approach of representing a text document is the Vector Space Model (VSM) [8], where a text document is modeled as elements in a vector space. Each element is representing an index term that is most useful identifying the main theme of a document. Baeza-Yates and Ribeiro-Neto gave the definition of vector space model as follows [61]:

"For the vector model, the pair (k_i, d_j) represents the occurrence frequencies that term k_i in the document d_j. A weight $w_{i,j}$ associated with the pair (k_i, d_j) is positive and non-binary."

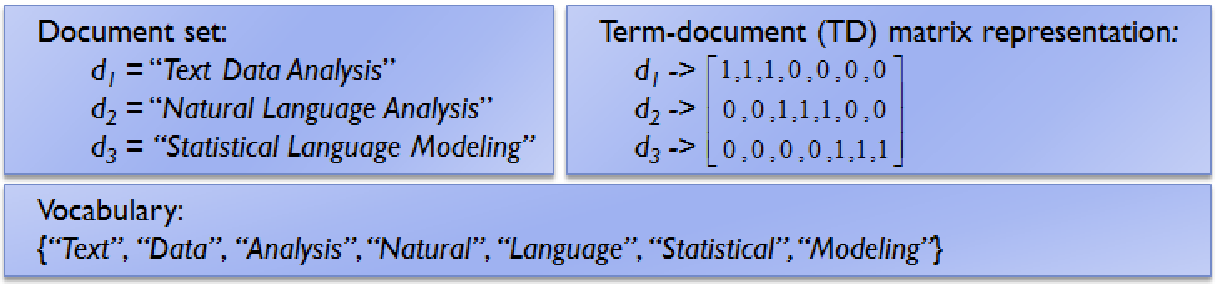
From the above definition, it is clear that VSM generation usually includes three stages: term selection, document indexing and weighting scheme selection.

In the first stage, text are tokenized into a stream of words, and a bag of content bearing terms, also known as indexed terms, are extracted from the document text. This term selection step, in the development of VSM, is considered as the most important step that largely impacts the system's performance. Generally speaking, it is done the words in the following several ways:

- Stop word removing: Words in a document do not describe the content which are called stop words (function words) are removed from document. The stop words can be identified with some automatic way. For example, terms which have very high or very low frequency can be considered as function words [62].

- Stemming: It is the process for reducing inflected words to their stem. For example, the words "accelerate", "acceleration", and "accelerating" are stemmed to the root word, "accel". As a result, it is considered as reducing dimension of selected words and help identifying similar words. The most widely used stemming algorithm is Porter Stemming algorithm [63]. The idea is that the suffixes in English mostly consist of a combination of smaller and simpler suffixes. In this dissertation, the stemming process is not our focus, considering that although stemming algorithm could reduce feature dimension, it also lose the information of full terms, and additional storage might be required to store both the stemmed and unstemmed forms [64].
- In the case of the classification problem, research works have been done in feature selection by using labeled training documents. This process ensures that the selected terms are highly related to the presence of a particular class, using a variety of measuring approaches such as Gini Index, Information Gain, Mutual Information, etc [65,66]. Again, this is not the focus in this dissertation, because that these measures are mostly based on training data, and important features for previously unseen documents may possibly be removed from this stage.
- In a lot of domain-specific applications in which text documents are usually unstructured, causally written, with plenty of grammar and spelling mistakes, as mentioned in section 1.2, misspelling correction is also a very essential step in removing noisy terms that could significantly help feature selection. Our work in this field will be presented in detail in Chapter 3.

The second stage is to construct a term-document (TD) matrix using indexed terms. Each entry in TD matrix indicates how many times that one term occurs in one document. More specifically, a document collection containing a total number of N documents identified by K terms is represented as a $K * N$ TD matrix. With the purpose of measuring how well each term describes the document contents, each entry of the TD matrix is represented by a "local weight" that is typically the occurrence frequency of individual term in one document [67]. Note here, each document is represented as a row vector in the TD matrix. An example of TD matrix generated from three documents is illustrated in Fig.4. To simplify the problem, all the words within these three documents are selected as indexed terms.



Document set:

d_1 = "*Text Data Analysis*"

d_2 = "*Natural Language Analysis*"

d_3 = "*Statistical Language Modeling*"

Term-document (TD) matrix representation:

$$\begin{array}{l} d_1 \to \\ d_2 \to \\ d_3 \to \end{array} \begin{bmatrix} 1,1,1,0,0,0,0 \\ 0,0,1,1,1,0,0 \\ 0,0,0,0,1,1,1 \end{bmatrix}$$

Vocabulary:

{"Text", "Data", "Analysis", "Natural", "Language", "Statistical", "Modeling"}

Fig. 4. Example of TD matrix generation

Research work has shown that using only local weight is insufficient to evaluate the importance of indexed terms [68]. For instance, some terms, due to their rare appearance in a few documents, do a better job to discriminate these documents from others. Some terms, on the contrast, appear too frequently in the whole collection to distinguish documents in different categories. As a result, after the TD matrix is generated, each entry needs to be weighted using a "global weight", which is used to reflect the overall importance of the index term in the whole document collection. The idea is that a term occurring rarely should have a high global weight and frequently occurring terms should be weighted low. Several well-known global weights are introduced in [69]:

Normal: $\sqrt{\frac{1}{\sum_j tf_{ij}^2}}$

GfIdf: $\frac{gf_i}{df_i}$

Idf: $\log_2\left[\frac{ndocs}{df_i}\right]+1$

1-Entropy or Noise: $1-\sum_j \frac{p_{ij}\log(p_{ij})}{\log(ndocs)}$ where $p_{ij}=\frac{tf_{ij}}{gf_i}$

tf_{ij} –occurrence frequency of term i within document j;

df_i– document frequency, which is the total number of documents in the document collection that contain term i.

gf_i– global frequency at which term i occurs in the entire document collection

ndocs – total number of documents in the whole document collection.

Generally speaking, each entry $w_{i,j}$ of TD matrix is assigned with two-part values, $w_{i,j}=L_{ij}*GW_i$, where L_{ij} is the local weight and GW_i is the term's global weight. The most commonly used term weighting scheme is *tf-idf* (term frequency-inverse document frequency) [67]. It assigns a high degree of importance to terms occurring rare in a document collection.

From the above discussion, when we take a deep look at those global weighting schemes, it is obvious that they are all focused on the entire document collection. Based on our observation, important term words or their synonyms appear frequently in documents in a specific category, especially when the user defined category is determined by some specific keywords [4]. Therefore, we designed a new global weighting scheme during the generation of VSM models, which will be discussed in detail in Chapter 4.

The advantage of using VSM model to represent text is that, instead of fully "understanding" the content of a document, the VSM simply generates a logical view of the document. For those applications with short, poorly-structured text documents or those documents that having grammar errors or spelling errors, VSM simplifies the procedure of natural language processing and ease the system implementation. These simplifications are proved effective by many research results [2,4,11,13,68]. This is the major reason that our work on text categorization is also using VSM as our fundamental text representation approach. However, VSM also has several limitations [70], such as poor representation of long documents, ignoring the semantic relationship between documents with similar context or synonyms, informal weighting schemes, etc. As a result, this dissertation presents our great effort in improving the conventional VSM model.

Considering the fact that traditional VSM can be inaccurate under the condition that a given word is expressed in many ways (synonymy) or a word has multiple meanings (polysemy) with different context, approaches that dig out underlying semantic relationships between words are required in many text mining applications. Latent semantic indexing (LSI) is one of the earliest techniques that try to solve the above problem.

2.4.2 Latent Semantic Indexing (LSI)

LSI algorithm, also called latent semantic analysis (LSA), is first introduced in 1988 for natural language processing [71]. It is a variant of the VSM that allows the low-rank approximation to the original TD matrix.

In the LSI each document is mapped into a lower dimensional space by decomposition of the TD matrix. The assumption made by LSI is that, some "latent" semantic structure exists according to the overall pattern of term occurrence. The low-rank approximation aims at merging the dimensions associated with similar terms, as well as enhancing or eliminating polysemy relationships based on right word meaning used in the context. The lower dimensional space is build using singular value decomposition (SVD) which is associated with the latent semantic structure [72]. The major steps of LSI are presented as following:

Given a $m \times n$ matrix A, it is decomposed into the products of three matrices by using SVD method.

$A = U\Sigma V^T$, where $U^T U = V^T V = I_n$, I_n is a size n identity matrix, $\Sigma = diag(\sigma_1, \sigma_2, ..., \sigma_n)$, $\sigma_i > 0$ for $1 \le i \le r$, and $\sigma_j = 0$ for $r+1 \le j \le n$, r is the rank of A, U and V contains left and right singular vectors of A, respectively, and diagonal matrix Σ contains the singular values of A. Then we get new matrices U_k ,V_k by keep only the k largest singular values of Σ, a rank-k approximation matrix to A is constructed with the following formula:

$A \approx A_k = U_k \Sigma_k V_k^T$.

The mathematical representation of SVD is shown in Fig. 5.

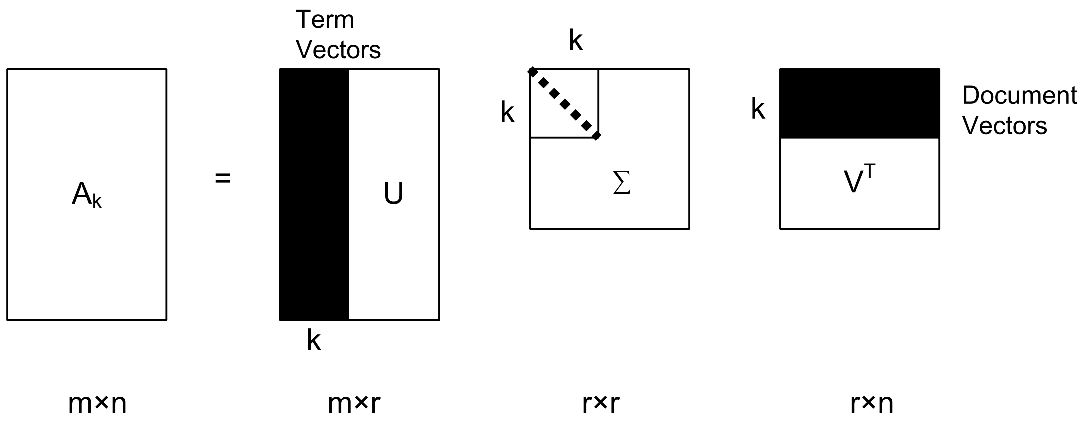


Fig. 5. Mathematical Representation of Singular Value Decomposition

The above SVD method attempts to capture underlying semantic structure in the association of terms and documents, as well as reducing feature dimensions by defining a much smaller value k than the number of indexed terms. It greatly reduces the memory requirement and the computing time of measuring the similarity between an unseen document and a known document. However, in this dissertation, we do not use this approach to capture the semantic relationship in text, due to the following drawbacks of LSI:

- The core technique of LSI, SVD, is a mathematical method; therefore the resulting singular values are not interpretable.
- SVD is very sensitive to the change of data. Sometimes the low-rank approximation has negative values which are meaningless [72].
- LSI requires relatively high computational performance and memory in comparison to other information retrieval techniques. Without distributed implementation it is not applicable to large-scaled document corpus [73].
- It still remains to be verified experimentally whether the LSI outperforms the VSM in text mining tasks such as text categorization. The concept in the LSI reduced-dimension space is assumed to be a weighted average of multiple meanings, while losing single term information and some real meaning information [68]. Under such cases, the LSI might exhibit a less satisfying classifying ability.

2.4.3 Statistical topic models

We now move from LSI to the discussion of statistical topic models, which is derived from LSI but interpret LSI from statistical point of view to provide a better understanding for text data. This is a major focus of improving conventional VSM model in this dissertation, since it provides a solid statistical foundation for finding hidden semantic structure of text documents.

2.4.3.1 Probabilistic Latent Semantic Analysis (PLSA)

Probabilistic latent semantic analysis (PLSA) is a well-known statistical topic model for text clustering and information retrieval [74]. It represents a document with a mixture distribution over "latent topics", which are characterized by a distribution over the indexed terms. The latent topics provide a reduced dimension representation of documents in a given collection. It is a statistical variant of LSI developed based on a statistical generative model called Aspect Model [74]. The starting point of PLSA is the term-document frequency (TDF) matrix, and it follows the bag-of-words assumption, in which each word appears independently, and the occurring order of each word is not considered. Fig. 6 shows the graphical model representation of PLSA, based on Bayesian Networks [109].

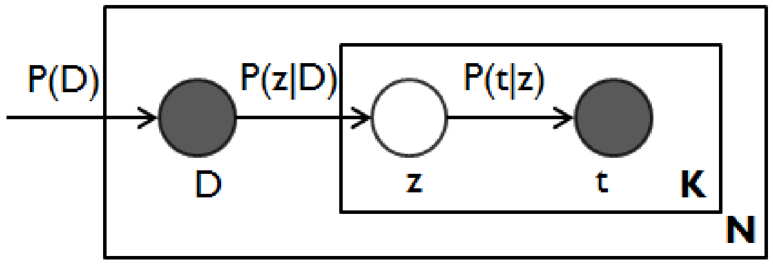


Fig. 6. Graphical model representation of PLSA

In the above graphical model, the solid circles *D* and *t* represent a document and a term that are observed respectively. The PLSA model is a generative model that assumes there is a latent "topic" variable *z* between documents and terms. The two rectangles, marked by *K* and *N*, represent the number of sample words and documents observed, respectively. $P(D)$, $P(z \mid D)$,

$P(t|z)$ represents the probabilities of observing a document *D*, a latent topic *z* occurring in *D*, and word *t* belonging to *z*, respectively.

The typical approach of PLSA modeling, is to estimate the probability functions $P(z|D)$ and $P(t|z)$, for all document-topic and term-topic pair (*z*, *D*) and (*t*, *z*) through machine learning. For each pair of document-topic, (z, D), and term-topic, (t, z), we attempt to find the values for functions $P(z|D)$ and $P(t|z)$ that maximize the following log–likelihood objective function:

$$L = \sum_{D,t} n(D,t) \log(P(D) \sum_{z} P(t|z)P(z|D)), \text{ (1)}$$

where $n(D,t)$ denotes the term frequency of *t* appears in document *D*.

The variables $P(z|D)$ and $P(t|z)$ are what we are interested in and want to estimate, since $P(D)$ is not related to the parameter we want to estimate and we assume that it is constant among documents in *Tr*, we then have:

$$\arg\max(L) \propto \arg\max \sum_{D,t} n(D,t) \log(\sum_{z} P(t|z)P(z|D)) \quad (2)$$

We will use the well-known Expectation Maximization (EM) algorithm [75] to solve this maximization likelihood estimation problem. Each iteration of EM algorithm consists of expectation step (E-step) and maximization step (M-step). In E-step, based on the current estimated $P(z|D)$ and $P(t|z)$, the posterior probability of $P(z|D,t)$ is computed for each document-word pair. In M-Step, $P(z|D)$ and $P(t|z)$ are updated by maximizing equation (2). This is an unsupervised machine learning process, and the detailed steps of EM algorithm will be discussed in Chapter 4.

2.4.3.2 Latent Dirichlet Allocation (LDA)

Similar with PLSA, Latent Dirichlet Allocation (LDA) is a generative probabilistic model for text collection [78]. In LDA, each document is modeled as a finite mixture over a set of latent topics. The basic idea is that documents are represented as random mixtures over latent topics, where each topic is characterized by a distribution over words [78]. LDA assumes the following generative process for each document *D* in a corpus *Tr*:

1. Choose $K \sim Poisson(\xi)$ for each of the *N* documents in *Tr*.
2. Choose $\theta \sim Dir(\alpha)$
3. For each of the *K* words in *D*,
    a. Choose a topic $z_n \sim Multinomial(\theta)$
    b. Choose a word t_n from $p(t_n | z_n, \beta)$, a multinomial probability conditioned on z_n.

There are several assumptions to be made in this model. First of all, the latent topic dimensionality *G* of the Dirichlet distribution is known and fixed. Secondly, the word probabilities are determined by a *G* * *K* matrix β where. Finally, *K* is independent of all the other data generating variables (θ and *z*) [78]. The *G*-dimensional Dirichlet random variable θ has the following probability density function:

$$p(\theta | \alpha) = \frac{\Gamma\left(\sum_{i=1}^{G} \alpha_i\right)}{\Pi_{i=1}^{G} \Gamma(\alpha_i)} \theta_1^{\alpha_1 - 1} \ldots \theta_G^{\alpha_G - 1},$$

where α is a size G vector with $\alpha_i > 0$, and $\Gamma(x)$ is the Gamma function. Given the parameters α and β, the joint distribution of a topic mixture θ, a set of K topics z, and a set of K words t is given by:

$$p(\theta, z, t | \alpha, \beta) = p(\theta | \alpha) \prod_{n=1}^{N} p(z_n | \theta) p(t_n | z_n, \beta)$$

Therefore, the LDA model is represented as a probabilistic graphical model in Fig. 7. It is obvious that LDA is a three-level hierarchical Bayesian model.

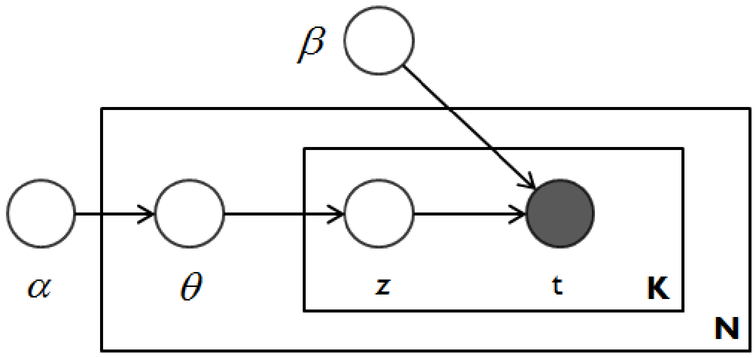


Figure 7. Graphical model representation of PLSA

Although LDA is claimed to overcome some shortcomings of PLSA, such as reducing number of parameters being estimated, and treating the topic mixture weights as a hidden random variable rather than a large set of individual parameters linked to training set [78], in this dissertation, we mainly focuses on extending basic PLSA model rather than applying LDA, based on the following several reasons:

- PLSA approximation is based on maximum likelihood estimation, while LDA is based on Bayesian estimation, by using both prior knowledge and available data [77]. When data size is large, their performance tends to be very similar [76].

- Considering the complexity of variational inference of LDA, PLSA is much easier to be implemented and extended using semi-supervised manner. We would rather focus on how to combine statistical topic model with supervised information from training set as well as external ontology resources, than find out which model or estimation method is better.
- With parameters partially fixed after training, PLSA could also be applied to process previously unseen document, thus makes text categorization possible. This will be further discussed in Chapter 4.

The following section will mainly discuss research works have been done in the field of text categorization, which is one of the major focuses in this dissertation.

2.5 Text categorization

Text categorization, as introduced in section 1.1, is one of the most popular tasks in text mining field nowadays, due to the increased availability of documents in digital form and the ensuing need to organize and differentiate them for further analysis. Mathematically, as described in [79], if Tr is a set of N documents, $Tr = \{D_1, D_2, ..., D_N\}$, and C is a set of predefined categories, $C = \{c_1, c_2, ..., c_M\}$, the task is to approximate the classifier that maps each $d_i \in Tr$ to a $c_j \in C$, so that the estimated target mapping function $\hat{\phi} : Tr \rightarrow C$ coincide the real mapping function $\phi : Tr \rightarrow C$ as much as possible.

Document categories defined by users always vary by different application requirements, such as the topic a news article discusses, the importance of a vehicle diagnostic record that describes vehicle repair details [80], the fact stated in a medical diagnostic document that whether or not an injury condition sustains [4], etc. One way to solve the text categorization problem is to use human experts to manually classify documents. Of course, this approach is costly and time-consuming. In the research community today, the dominant approach to this problem is based on machine learning techniques, which is proved to be effective in analyzing large amount of data and has the straight portability to different application domains [81].

2.5.1 Text categorization based on machine learning algorithms

Machine learning approach for text categorization has gained popularity since early 90's. In this approach, a general inductive process is adopted to develop a classifier that classifies previously unseen documents by gleaning the characteristic of available training documents. There are already plenty of research works that focus on developing and improving machine learning techniques adopted for building classification models, which are generally reviewed in [76,81]. A list of major techniques that have been applied in text categorization literature is presented as following:

- Classifiers based on document clustering algorithms such as k-means clustering [82], hierarchical clustering [83], self-organizing maps [4,84], etc.
- Example based classifiers such as k-nearest-neighbor classifiers [92,93] that classify unseen data by finding the closest data samples in the training set using similarity measures.
- Probabilistic classifiers such as Naïve Bayes classifiers [85,86] that measures the probability of a sample belongs to a certain category.
- Decision tree classifiers, which is a hierarchical decomposition of the training data space. At each node of the tree, the attribute that most effectively splits data samples into respective subsets is selected, based on information gain ratio. The splitting is recursively conducted until the leaf nodes contain a certain minimum number of samples, or some conditions on class purity are met [76].
- Classifiers that are derived from regression related algorithms, such as Linear Least Squares Fit (LLSF) method [87], logistic regression classifier [88], neural network classifier which is based on logistic regression and usually is considered as a nonlinear combination of a number of logistic regression classifiers [89,90,91].

- Linear classifiers such as support vector machines (SVM) [94,95], which is a type of classifiers that attempt to determine “good” linear separators among categories. It is proposed first by Vladimir Vapnik in 1979, but did not receive much attention until late 90’s. The basic idea is to find out a separation hyperplane for data samples so that the normal distance of any of the data points from the hyperplane is the largest. Fig. 8 shows an example of 2-dimensional case for SVM classifier learning. The crosses and circles represent training examples in different categories, whereas lines represent decision surfaces, and the thicker line represents the best separation hyperplane, since the distance from it to the nearest data point is maximized. Small squares indicate the support vectors, which are training samples lying on the maximum margin surface [81].

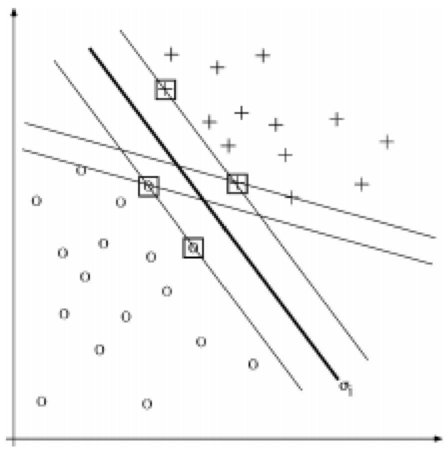


Figure 8. Graphical model representation of PLSA

In this dissertation, we consistently use SVM as our text categorization classifier throughout different experiments, with the following major reasons:

- SVM provides much more robust performance as compared to many other machine learning techniques such as rule based classifiers and decision trees [97].

- SVM is quite robust to high dimensionality. It is ideally suited for text categorization because of the sparse high-dimensional nature of text [96].
- Term selection is often not needed in SVM classification, as SVMs tend to be fairly robust to over-fitting and can scale up to considerable dimensionalities [76].
- Whatever machine learning techniques are used to build text classifiers, the classification accuracy will be "bottlenecked" if the representation quality of the document is poor, i.e., the representation of a document does not reflect close relationship with its assigned category. Furthermore, based on [98], research works have proved that the sophistication of feature selection in text categorization is more important than choosing the best classifier. Therefore, our research focus in this dissertation on the text categorization task is rather improving text representation and combining with a consistent and promising machine learning approach for evaluation, than choosing and improving the classifier itself.

2.5.2 Text categorization based on statistical topic models

Statistical topic models, including PLSA, LDA, etc., provide a solid probabilistic foundation for document modeling and representation, in terms of digging out latent semantic structures from text. In the literature, these models are mainly used for unsupervised text clustering, information retrieval and dimension reduction, and research works of utilizing them for text categorization is still rare. Most of the work been done on applying topic model to text categorization is to purely use estimated latent features for classification, such as [99,100,101,102], which is claimed in [103] to be less accurate than using bag-of-word (BOW) features, especially when training data size is large. The major limitation of the above work is that, text categorization requires a sophisticated feature selection approach that generates a well-developed text representation. Using semantic or syntactic structure learned from text is not sufficient in fully representing the content of the document, and different application requirements for text categorization make it essential to develop a hybrid classifier with enriched features from multiple resources such as single words, relationship between words, semantic structure, word context, etc.

2.5.3 Text categorization based on ontology network

Considering the necessity of improving BOW features for more accurate text categorization, as mentioned in Section 1.2, researchers have been working on utilizing external or background knowledge to help build text classifier [104,105]. These external knowledge has a great advantage in helping extract semantic relationships, match important phrases, strengthen co-occurrences, etc. As discussed in section 2.3, WordNet is one of the best known sources of external knowledge used for text categorization. It is a large lexical database of English first developed and maintained by Princeton University [49]. The main relation among words in WordNet is synonymy, e.g., as between the words shut and close or car and automobile. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept, together with short explanations and general definitions. For the purposes of text categorization, it is successfully used to unify the vocabulary across the documents by modifying the document features with use of the related words. The key rules defined in the most well-known and widely used approach in [46] are listed as follows:

- Add rules: extend each row vector $\vec{W}_D$ in TD matrix defined in section 2.4.1 by adding new entries for WordNet concepts c appearing in the document set. $\vec{W}_D$ is replaced by the concatenation of $\vec{W}_D$ and $\vec{C}_D$, where $\vec{C}_D = [cf(D,c_1), cf(D,c_2)...cf(D,c_V)]$, where $cf(D,c)$ denotes the frequency that a concept *c* appears in document *D*, and *V* denotes the total number of concepts generated.

- Replace: Each row vector $\vec{W}_D$ in TD matrix is replaced by the concatenation of $\vec{W}'_D$ and $\vec{C}_D$, where $\vec{W}'_D$ denotes the new vector after removing all term entries from $\vec{W}_D$ that appear in WordNet. Terms that do not appear in WordNet are not discarded.
- Concept only: remove all terms from the vector representation that do not appear in WordNet. Only $\vec{C}_D$ is used to represent document *D*.

It is obvious that the above rules have plenty room of improvement. Several open issues include:

- Which rule/combination of rules should be selected to apply to the document features?
- What value should be assigned to the concept features added to TD matrix after global weighting scheme for indexed terms is applied?
- How to determine the scope of synset with multiple meanings, for a word with multiple word categories in a document?
- How to make full use of word relationships other than synonymy (e.g., Hypernym/hyponym)?

These problems discussed above are important indications that text categorization using ontology networks is worth further investigation and perfection. Therefore, it is one of the major focuses in this dissertation, and will be discussed in detail in Chapter 4, with proposed solutions.

CHAPTER 3. AUTOMATIC TYPO CORRECTION USING MACHINE LEARNING AND EXTERNAL KNOWLEDGE BASES

In this chapter, we present our research in typo correction for processing unstructured documents with a focus on three types of non-word typos: word boundary errors, self-invented abbreviations, and ambiguous acronyms. We present an innovative automatic typo correction system, ITDC (Intelligent Typo Detection and Correction), that uses hybrid knowledge, i.e. general language lexicon and domain-specific knowledge extracted through machine learning. We developed algorithms for misspelling detection and correction candidate generation, approximate word matching, topographically similar word grouping, extracting contextual knowledge from general and application domains, and candidate ranking based on machine learning and statistical analysis. The corrected text documents can then be used for further text processing, such as text document categorization and text document retrieval.

3.1 Machine learning algorithms to extracting knowledge for typo correction

Automatic typo detection and correction is a challenging problem, in particular in unstructured text documents that contain acronyms, abbreviations and symbols that are specific to application domains. We propose to use a hybrid of knowledge, general language specific typo knowledge, and domain specific knowledge. Two general language knowledge bases, GTKB1 (General Typo Knowledge Bases), and GTKB2 are important for typo detection and corrections. GTKB1 is a valid word lexicon that can be automatically generated from online English dictionaries such as WinEdt English_US and/or English_UK [20]. GTKB1 can be used for typo detection and generating typo correction candidates. The most challenging part in typo processing is to correct detected typos, which can be either typographical errors or non-word terms that can were used as abbreviations or symbols meaningful only to a specific application domain. To correct typographical errors, a general language knowledge base of commonly misspelled words, GTKB2, can be automatically generated from internet sources including Wikipedia Common Typo List, Oxford English Corpus Misspelling List, and Student's Book of College English Misspelling list and [39-41]. The focus of our research is to explore the use of machine learning technologies to extract domain specific knowledge that are useful for correcting challenging typos, such as word boundary errors, self-invented or domain specific acronyms and abbreviations. Fig. 9 gives a summary of the knowledge useful for typo correction. In this section we introduce the machine learning algorithms developed to extract domain specific typo knowledge from text corpus collected within a specific domain, including true words for the acronyms and abbreviations, topographically similar words/typos groups, and typos due to wrong word boundaries.

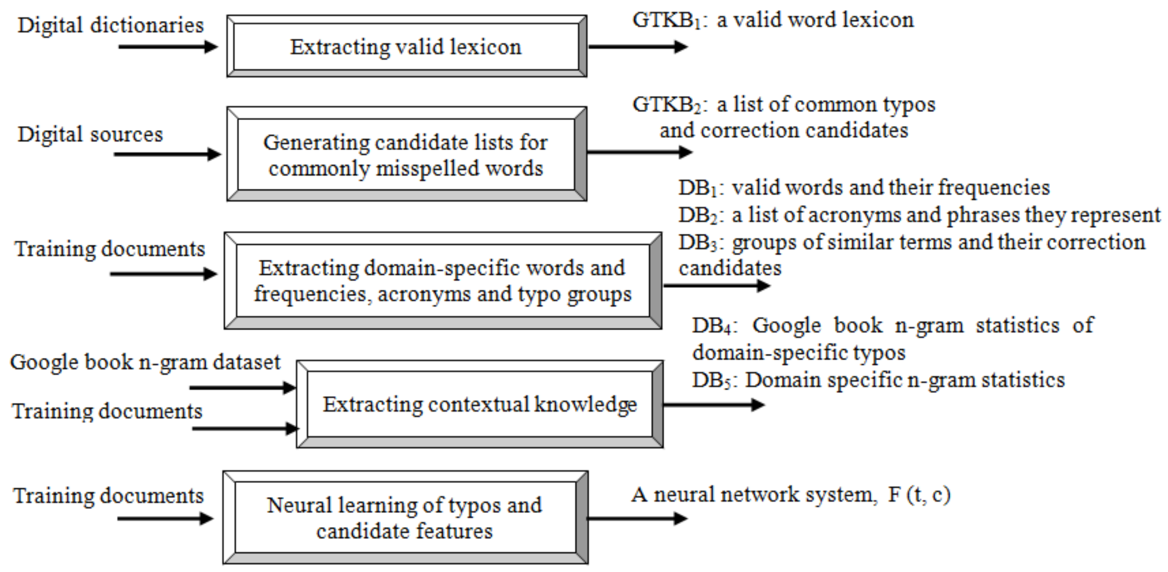

Digital dictionaries
Extracting valid lexicon
GTKB1: a valid word lexicon
Digital sources
Generating candidate lists for commonly misspelled words
GTKB2: a list of common typos and correction candidates
Training documents
Extracting domain-specific words and frequencies, acronyms and typo groups
DB1: valid words and their frequencies
DB2: a list of acronyms and phrases they represent
DB3: groups of similar terms and their correction candidates
Google book n-gram dataset
Training documents
Extracting contextual knowledge
DB4: Google book n-gram statistics of domain-specific typos
DB5: Domain specific n-gram statistics
Training documents
Neural learning of typos and candidate features
A neural network system, F (t, c)

Fig. 9. Extracting knowledge for typo detection and correction

3.1.1 Extracting knowledge of domain-specific terms and acronyms

We propose to build an enhanced domain-specific dictionary that can used to identify valid terms commonly used in an application domain, and acronyms that are valid and represent words meaningful within an application domain. For example, in automotive diagnostic applications, NPF is a common acronym known as “No Problem Found”, and CAN known as Controller Area Network, which can be ambiguously interpreted.

From a given training data set we first generate a list of valid words and their frequencies occurring in a training data set, which is denoted as DB_1. We developed a semi-automatic process to extract a list of acronyms from a given corpus of training documents collected from a specific application domain. It first extracts all typos with short lengths from the training documents. For each of these typos, we search for the valid phrases in the training set that can form the typo with its initials. For example, for the typo “NPF”, we searched in the training documents and found the phrase “No Problem Found” occurring 5 times, “Noise Possibly From” 5 times, and “Not Put Fuel” 3 times. The extracted phrases are then ranked according to their frequencies of occurrences. The phrase with the highest frequency of occurrences is assigned to the typo as the correction candidate. If there is a tie, such as the example given above, a domain expert manually identifies the best correction candidate. This knowledge base is denoted as DB_2:

$$DB_2 = \{(ac_1, ex(ac_1)), (ac_2, ex(ac_2)), \ldots (ac_H, ex(ac_H))\}. \qquad (1)$$

Where ac_i denotes an acronym and $ex(ac_i)$ denotes the correct phrase ac_i represents.

3.1.2 Building a lexicon of similar typos and domain-specific abbreviations

One of the characteristics of unstructured text in a specific application domain, for instance, vehicle diagnostic text records, is that they contain many abbreviations and casual word patterns that cannot be easily corrected by only using common English dictionaries. For example, “wheel” -> “whl”, “check” -> “chk”, “diagnose” -> “diagn”, etc. As a result, it is necessary to build a domain-specific lexicon [22] to correct these word errors. The following is an algorithm we developed to build a domain-specific lexicon, which is represented in groups of words and typos such that the words and typos in each group share the same stem word. For example, typos “cusomer”, “cutomer”, “custmer”, “customers”, “cusotmer” and “customer” all shares the same stem word “custom”.

The algorithm uses four similarity measures to detect four basic types of typing/spelling errors, deletion (Type 1), insertion (Type 2), substitution (Type 3) and transposition (Type 4). Research showed that more than 80% of typos belong to one of these four error types [38]. Suppose *A* and *B* are two words, $A = A_1A_2...A_n$ is a typo and $B = B_1B_2...B_m$ is a valid word, where A_i and B_j, $i = 1,2,...n; j = 1,2,...m,$ are letters within the alphabet. If *A* is formed by deleting one or more letters from *B*, such as: *A* = ENINE, and *B* = ENGINE, then *A* belongs to Type 1 topographical spelling error. If *A* is formed by inserting one or more letters into *B*, such as: *A* = FOUOND, and *B* = FOUND, then *A* belongs to Type 2. If *A* is formed by substituting one or more letters with wrong letters, such as: *A* = STSTEM, and *B* = SYSTEM, then *A* belongs to Type 3. If *A* is formed by transposing two letters in *B*, such as: *A* = PERFROM, and *B* = PERFORM, then *A* belongs to Type 4. Fig. 10 illustrates the four types of misspelling words.

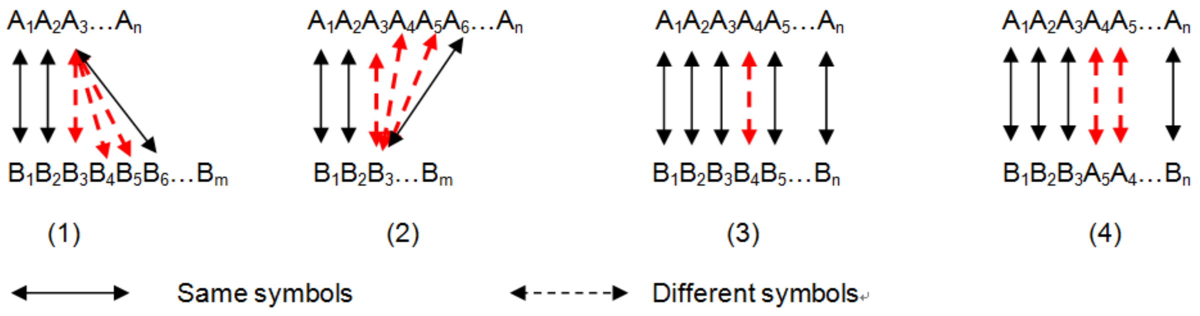


Fig. 10. Four types of topographical spelling errors

We designed the following algorithm to calculate the distance between two words, *A* and *B*, based on the above four types of spelling error.

- Similarity measure calculated based on type 1 spelling errors:

  1) Let $s_1 = 0$. Starting from the first letter of *A* and *B*, $x = 1, y = 1$.

  2) Compare A_x and B_y,

    2.1) If $A_x = B_y$, then $s_1 = s_1 + 1$, and $x = x + 1, y = y + 1$.

    2.2) If $A_x \neq B_y$, then go to the next letter of *B*, $y = y + 1$.

    2.3) If $x \leq n$ and $y \leq m$, go to step 2.1, otherwise exit.

- Similarity measure calculated based on type 2 spelling errors:

    1) Let $s_2 = 0$. Starting from the first letter of *A* and *B*, $x = 1, y = 1$.

    2) Compare A_x and B_y,

    2.1) If $A_x = B_y$, then $s_2 = s_2 + 1$, and $x = x + 1, y = y + 1$.

    2.2) If $A_x \neq B_y$, then go to the next letter of *A*, $x = x + 1$.

    2.3) If $x \leq n$ and $y \leq m$, go to step 2.1, otherwise exit.

- Similarity measure calculated based on type 3 spelling errors:

    1) Let $s_3 = 0$. Starting from the first letter of *A* and *B*, $x = 1, y = 1$.

    2) Compare A_x and B_y,

    2.1) If $A_x = B_y$, then $s_3 = s_3 + 1$, and $x = x + 1, y = y + 1$.

    2.2) If $A_x \neq B_y$, then go to the next letter of both *A* and *B*, $x = x + 1, y = y + 1$.

    2.3) If $x \leq n$ and $y \leq m$, go to step 2.1, otherwise exit.

- Similarity measure calculated based on type 4 spelling errors:

    1) Let $s_4 = 0$. Starting from the first letter of *A* and *B*, $x = 1, y = 1$.

    2) Compare A_x and B_y,

    2.1) If $A_x = B_y$, then $s_4 = s_4 + 1$, and $x = x + 1, y = y + 1$.

    2.2) If $A_x \neq B_y$, $A_{x-1} \neq B_{y-1}$ and $A_x = B_{y-1}$, $A_{x-1} = B_y$, then $s_4 = s_4 + 1$, and $x = x + 1, y = y + 1$.

    2.3) Else if $A_x \neq B_y$, then go to the next letters in both *A* and *B*, $x = x + 1, y = y + 1$.

    2.4) If $x \leq n$ and $y \leq m$, go to step 2.1, otherwise exit.

- The distance between two terms A and B is then calculated as:

$$\text{Dist}(A,B) = 1 - \left(\frac{s}{\max(|A|,|B|)}\right), \qquad (2)$$

where $s = \max\{s_i | i = 1,2,3,4\}$.

For example, for terms PEEFROM and PERFORM, we have similarity measures s_1, s_2, s_3 and s_4 equal to 2, 3, 4 and 5, respectively. As a result, $s = 5$, which means the best matching case of typo "PEEFROM" is type 4. The term distance measure function, *Dist*, is used in the following algorithm for grouping similar terms, so the typographically similar terms, which can be valid words or typos, are placed in the same term group with the same correction candidate.

Typo correction algorithm based on grouping similar terms

Let D be a set of training documents.

1) Extract a list of distinct terms from D, denoted as T.

2) Generate groups of term words in T that are topographically similar. The following steps find the topographically similar terms in T and generate groups of similar terms, $\{g_1, g_2, \ldots, g_K\}$.

2.1) Let $j = 1, k = 1$.

2.2) Create a new group g_k. Take the term t_j from term list T, and add it into group g_k.

2.3) Increment j. If $j > \|T\|$, go to step 3.

2.4) Take the term t_j from term list T.

2.5) If $length(t_j) \le \tau_l$, where τ_l is a threshold used for filtering out short terms, go to step 2.3.

2.6) Calculate μ_i, the average distance between t_j and each word t_q in each existing term group g_i.

$$\mu_i = \frac{\sum_{q=1}^{Q_i} Dist(t_{j+1}, t_q)}{Q_i}, t_q \in g_i, q = 1,2,...Q_i; i = 1,2,...\eta. \quad (3)$$

Where η is the number of the currently existing groups, and Q_i denotes the number of terms in group g_i.

2.7) Find the closest group g_C to t_j , i.e. $\mu_C \le \mu_i$ for all *i*, $1 \le i \le \eta$.

2.8) If $\mu_C < \tau_\mu$, where τ_μ is the maximum distance allowed between the terms of the same term group, add t_{j+1} into g_C, and go to step 3. Otherwise, increment *k*, add t_j to g_k and go to step 2.3.

3) Assigning a correct word to each group.

3.1) Let $i = 1$,

3.2) If $i > k$, output:

$$DB_3 = \{(g_1, g\_w_1), (g_2, g\_w_2), ...(g_k, g\_w_k)\} \quad (4)$$

And then exit.

3.3) If there are valid words in g_i, find a valid word in g_i that has the highest frequency of occurrences in training data *D*, and denote the word as g_w_i. Add (g_i, g_w_i) to DB_3, and go to step 3.5.

3.4) Find a valid word in *T* that has the closest distance go each term in g_i, assign the word to g_w_i, and add (g_i, g_w_i) to DB_3.

3.5) Increment *i* and go to step 3.2.

Table 1 shows the two word/typo groups generated by the above algorithm from a set vehicle fault diagnostic text documents. For instance, topographically similar terms such as

“ENGI”,”ENGNIE” are grouped together and labeled as the most frequently occurred valid word “ENGINE”.

Table 1 EXAMPLE OF SIMILAR TERM GROUPS

Group sample	Elements	Label
1	ACCELERATION, ACCELLERATE, ACCELLERATING, ACCELLERATION, ACCELLING, ACCELING, ACCELORATION, ACCELRATE, ACCELRATION	ACCELERATE
2	ENGI, ENGIEN, ENGIN, ENGINE, ENGINES, ENGING, ENGINR, ENGNIE, ENGNINE, ENIGINE, ENIGNE, ENINE,	ENGINE

3.1.3 *Extracting contextual knowledge*

Contextual information such as n-gram has been found useful in detecting real-word errors [24, 25]. An n-gram is defined as a sequence of *n* adjacent words from a given text. A 1-gram is referred to as a "unigram"; 2-gram is a "bigram"; 3-gram is a "trigram". Google has been using n-gram word models in a variety of projects, including statistical machine translation, misspelling correction, entity detection, information retrieval, etc. We developed algorithms to extract two types of contextual knowledge and use the knowledge to rank the correction candidates for a given typo. The first type is generated based on the Google book n-gram corpus [23], which provides the frequencies of words and phrases appearing in American English books published from 1500s to 2000s. The second type of contextual knowledge is represented by the probabilities of n-grams occurring in the training documents *D*.

Algorithm for extracting contextual knowledge from Google book n-gram

1) Detect and extract all typos from the training document set *D* through dictionary search, and store them in a list, *D_typos*.

2) For each typo *t* in *D_typos*,

2.1) For n = 3 to *G*,

- Extract three types of n-grams, prefix n-grams, suffix n-grams and centered n-grams from training data *D*:

  $ngram_{pre}(t) = R_{n-1}$, such that $R_{n-1} \parallel t \in D$, where R_{n-1} represents *n*-1 words immediately precede *t*.

  $ngram_{suf}(t) = S_{n-1}$, such that $t \parallel S_{n-1} \in D$, where S_{n-1} represents *n*-1 words immediately follow *t*.

  $ngram_{cen}(t) = \{R_m, S_m\}$, such that $R_m \parallel t \parallel S_m \in D$, where R_m represents *m* words immediately precede *t*, S_m represents *m* words immediately follow *t*, where $2*m+1=n$. Note here, if either R_m or S_m is empty, then $ngram_{cen}(t)$ is empty.

- Extract $R_{n-1} \parallel *, * \parallel S_{n-1}$ and $R_m \parallel * \parallel S_m$ from Google book n-gram corpus, where * represents any valid word.

2.2) The n-grams generated by step 2.1 are denoted as ng_i, $i = 1,2,\ldots,L$. For each ng_i, we extract its normalized frequency of occurrences in the Google book n-gram corpus using the following formula:

$$H(ng_i) = \sum_{j=1}^{Y} \alpha_{yr_j} \left(\frac{f_{yr_j}(ng_i)+1}{total\_count_{yr_j}}\right) \qquad (5)$$

Where $j = 1,2,...Y$, Y represents the number of years of the Google book n-gram are used, yr_Y represent the most recent year and yr_1 represent the most least recent year in the last Y years. $f_{yr_j}(ng_i)$ represents the normalized frequency of ng_i appears in year yr_j in the Google book n-gram corpus. Here, $f_{yr_j}(ng_i)$ is added by 1 to avoid 0 values. $total\_count_{yr_j}$ represents the total count of words appear in year yr_j, and α_{year} is a weight coefficient that can be used to give different weights to different years, e.g., one can assign higher weights to n-gram frequencies of more recently years by using the exponential function,

$$\alpha_{year} = e^{-(\frac{yr_Y - yr_j}{yr_Y - yr_1})} .$$

The output from the above step is presented as knowledge base, DB$_4$:

$$DB_4 = \{(ng_1, H(ng_1)), (ng_2, H(ng_2)), ...(ng_L, H(ng_L))\}. \qquad (6)$$

We will use an example to illustrate the contextual knowledge generating process. Suppose for a typo x = "whele" occurred in a training document, and the training data set contained a 3-gram phrase "right front whele". We searched the Google book n-gram corpus from yr_1 = 1980 to yr_Y = 2008, and found 100 3-gram phrases, "right front *", where * represent a valid word. Examples of such 3-grams include "right front panel", "right front wall", "right front wheel", "right front where" and "right front window". Fig. 11 illustrates frequencies of occurrences of these n-grams in each year in the Google book n-gram corpus. It is obvious that the 3-gram phrase "right front wheel" occurred far more frequent than other phrases.

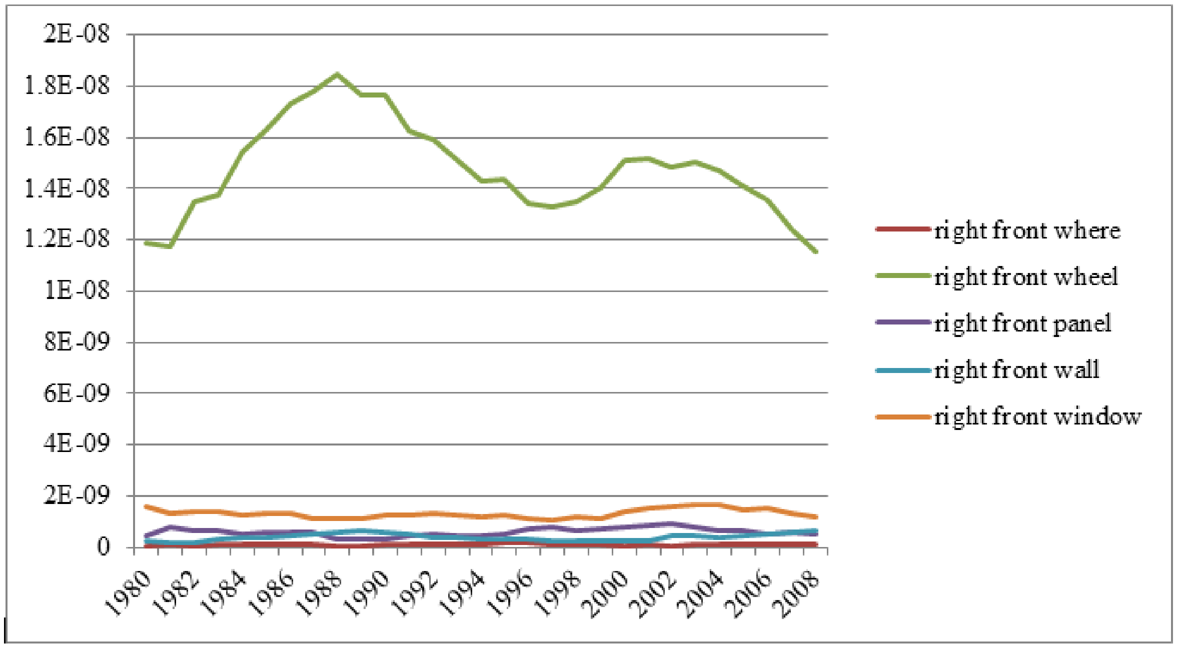


Fig. 11. Example of n-gram statistics

The second type of contextual knowledge is represented by the probabilities of n-grams occurring in the training documents *D*. Thus a knowledge base DB$_5$ is generated to contain domain-specific contextual information. Specifically, we extract three types of statistics for each valid word *x* in *D*:

- **Probability of prefix 2-grams (** $P_{2-gram}(Rx)$ **)**: This is the probability of the occurrences of all the 2-grams consisting of *x* and the preceding token *R*. For instance, for the word "front" in a 2-gram "left front", *R* is the word "left".
- **Probability of suffix 2-grams (** $P_{2-gram}(xS)$ **)**: This is the probability of the appearance of all the 2-grams consisting of *x* and the subsequent token *S*. For instance, for the word "front" in the 2-gram "front wheel", *S* is the word "wheel".
- **Probability of centered 3-grams (** $P_{3-gram}(RxS)$ **)**: This is the probability of the appearance of all the 3-grams consisting of the preceding token *R*, the word *x*, and the

subsequent token *S*. For instance, for the word “front” in the 3-gram “left front wheel”, *R* is the word “left”, *S* is the word “wheel”.

Using Bayes formula, we obtain:

$$P_{2-gram}(Rx) = P(x \mid R)P(R) = \frac{f(Rx)}{f(R*)} \times \frac{f(R)}{f(*)} \qquad (7)$$

$$P_{2-gram}(xS) = P(x \mid S)P(S) = \frac{f(xS)}{f(*S)} \times \frac{f(S)}{f(*)} \qquad (8)$$

$$\begin{aligned} P_{3-gram}(RxS) &= P(R \mid xS)P(x \mid S)P(S) \\ &= \frac{f(RxS)}{f(*xS)} \times \frac{f(xS)}{f(*S)} \times \frac{f(S)}{f(*)} \end{aligned} \qquad (9)$$

Where $f(*), f(R*), f(*S)$ denotes the frequency of all words in *D*, the frequency of any 2-gram in *D* that starts with *R*, and the frequency of any 2-gram in *D* that ends with *S,* respectively.

As a result, DB$_5$ is represented as:

$$DB_5 = \{\forall x \in T \mid \vec{P}(x) = (P_{2-gram}(Rx), P_{2-gram}(xS), P_{3-gram}(RxS))\} \qquad (10)$$

DB$_5$ is used to generate statistical features for the neural network trained for assessing typo correction candidates, which will be discussed in section 2.4.

3.1.4 Assessing typo correction candidates

In many cases, the closest correction candidate generated by topographical similarity matching may not be the correct one. The knowledge bases described above were generated based on various aspects of characteristics of typos, each one of the candidates generated based on these knowledge bases has a possibility to be the right correction. We developed a Neural Network system, NN_TC_Conf, to measure the confidence about a candidate word for a given typo over a broad range of typo-word features. Fig. 12 illustrates the architecture of the NN_TC_Conf system. The neural network uses the following ten features to characterize the weight of a correction candidate.

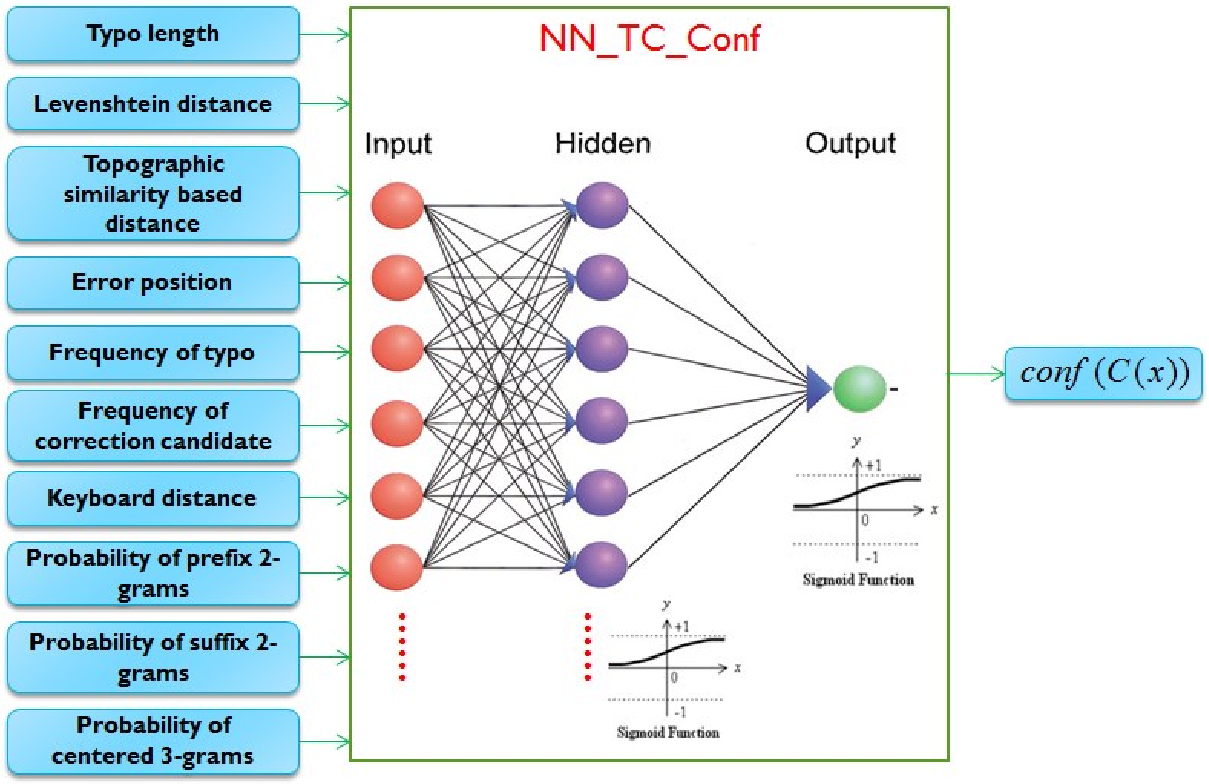

Typo length
Levenshtein distance
Topographic similarity based distance
Error position
Frequency of typo
Frequency of correction candidate
Keyboard distance
Probability of prefix 2-grams
Probability of suffix 2-grams
Probability of centered 3-grams
NN_TC_Conf
Input
Hidden
Output
Sigmoid Function
Sigmoid Function
$conf(C(x))$

Fig. 12. *NN_TC_Conf:* a neural network for measuring the confidence about a correction candidate of a typo

- **Typo length**: We chose this feature to take into consideration that the effect of typo length may have on its correction candidates. Based on our observation, the longer a typo is, less correction candidates it might have, but the possibility of a candidate being the correct word increases.
- **Levenshtein distance**: The second feature we considered is the Levenshtein distance [21] between the correction candidate and the typo, which is defined as follows. Let

$A = A_1A_2...A_n$ be a typo and $B = B_1B_2...B_m$ be a valid word, where A_i and B_j, $i = 1,2,...n; j = 1,2,...m$, are letters within the alphabet, and A_0 and B_0 denote *nil*. The Levenshtein distance between *A* and *B*, $L\_Dist(A,B) = L\_D(A_n, B_m)$, where $L\_D(A_i, B_j)$ is a recursive function defined as follows: for each *i, j*,

$$L\_D(A_i, B_j) = \begin{cases} 0, i = j = 0 \\ i, j = 0 \,\&\, i > 0 \\ j, i = 0 \,\&\, j > 0 \\ U, otherwise \end{cases}$$

$$U = Min(L\_D(A_{i-1}, B_{j-1}) + V, L\_D(A_{i-1}, B_j) + 1, L\_D(A_i, B_{j-1}) + 1)$$

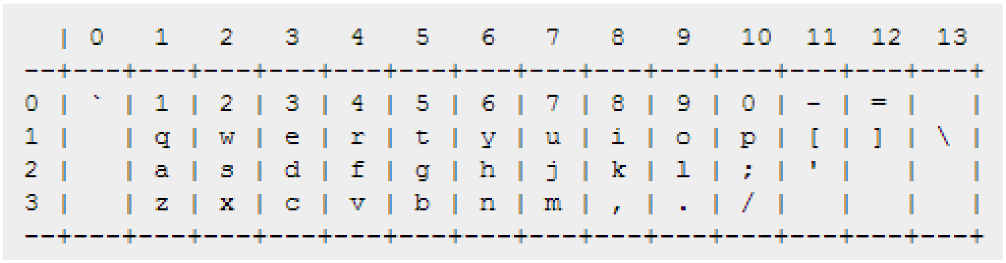
$$V = \begin{cases} 0, A_i = B_j \\ 1, otherwise \end{cases}$$

Where $L\_D(A_i, B_j)$ denotes the number of single-character edits (insertion, deletion, substitution) required to transfer sequence $A_1A_2...A_i$ to sequence $B_1B_2...B_j$. For example, for the typo word *A* = "engne", if the candidate *B* = "engine", the distance between *A* and *B*, $L\_Dist(A,B) = L\_D(A_5, B_6) = 1$, as shown in Table 2, because only one insertion edit of letter "i" is needed from *A* to *B*. Similarly, if the candidate *B* = "engineer", $L\_Dist(A,B) = L\_D(A_5, B_8) = 3$. As a result, "engine" is more likely to be the correct word for typo "engne." The smaller this distance is, the more likely the candidate is the correct word.

Table 2 EXAMPLE OF LEVENSHTEIN DISTANCE

		j=0	*j*=1	*j*=2	*j*=3	*j*=4	*j*=5	*j*=6
			e	n	g	i	n	e
i=0		0	1	2	3	4	5	6
i=1	e	1	0	1	2	3	4	5
i=2	n	2	1	0	1	2	3	4
i=3	g	3	2	1	0	1	2	3
i=4	n	4	3	2	1	1	1	2
i=5	e	5	4	3	2	2	2	**1**

- **Topographic similarity based distance:** This is the distance between a correction candidate, *A*, and the typo, *B*, calculated using the distance function *Dist(A, B)* introduced in (2) in section 2.2.
- **Error position**: This feature, denoted as *EP*, gives the position of the first error in the typo with respect to the correction candidate. Based on [7], this feature is important because studies have shown that errors were more frequent in certain positions. For a typo *A* and its correction candidate *B*, $EP = \frac{e}{|A|}$, where *e* is the index of the first letter that satisfy $A_e \neq B_e$.
- **Frequency of a typo**: This is the frequency of the typo occurring within the training document set, which can be extracted from DB_1.
- **Frequency of a correction candidate**: This is the frequency of the correction candidate occurred within the training document set, which can be extracted from DB_1. This feature is useful when a typo is actually an abbreviation or a term frequently used in the application domain. In this case, the typo's occurrence in the training documents may be high, but the suggested correct candidate may not occur very often.
- **Keyboard distance**: The value of this feature, denoted as *K_D*, is calculated based on the distance between keys on a keyboard using the QWERTY keyboard mapping [26], as shown in Fig. 13.



```
  | 0   1   2   3   4   5   6   7   8   9   10  11  12  13
--+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
0 | ` | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | - | = |   |
1 |   | q | w | e | r | t | y | u | i | o | p | [ | ] | \ |
2 |   | a | s | d | f | g | h | j | k | l | ; | ' |   |   |
3 |   | z | x | c | v | b | n | m | , | . | / |   |   |   |
--+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

Fig. 13. QWERTY keyboard distance matrix

For a typo *A* and a correction candidate *B*, the *K_D* is calculated as follows:

1) Calculate the four similarity measures, s_1, s_2, s_3 and s_4 between *A* and *B* as discussed in Section 2.2.

2) If $s_1 \geq s_i, i = 2,3,4$, then *A* is a deletion (Type 1) error.

2.1) Detect $B_{z+1}B_{z+2...}B_{z+\ell}$ within candidate *B* such that after deleting these letters from *B* we can obtain *A*.

2.2) Obtain the coordinates of $B_z, B_{z+1}, B_{z+\ell}, B_{z+\ell+1}$ based on Fig. 7. For example, if $B_z = "a"$, its coordinates are $co_{B_z} = (2,1)$.

2.3) Calculate the *K_D* value between *A* and *B*:

$$K\_D = \left|co_{B_z}, co_{B_{z+1}}\right| + \left|co_{B_{z+\ell}}, co_{B_{z+\ell+1}}\right| \qquad (11)$$

3) If $s_2 \geq s_i, i = 1,3,4$, then *A* is an insertion (Type 2) error.

3.1) Detect $A_{z+1}A_{z+2}...A_{z+\ell}$ in *A* such that after deleting these letters from *A* we can obtain *B*.

3.2) Obtain the coordinates of $A_z, A_{z+1}, A_{z+\ell}, A_{z+\ell+1}$ based on Fig. 7.

3.3) Calculate the *K_D* value between *A* and *B*:

$$K\_D = \left|co_{A_z}, co_{A_{z+1}}\right| + \left|co_{A_{z+\ell}}, co_{A_{z+\ell+1}}\right| \qquad (12)$$

4) If $s_3 \geq s_i, i = 1,2,4$, then *A* is a substitution (Type 3) error.

4.1) Detect $B_{z+1}B_{z+2...}B_{z+\ell}$ in *B* such that *B* will equal to *A* after substituting these letters in *B* by $A_{z+1}A_{z+2}...A_{z+\ell}$ from *A*.

4.2) Get the coordinates of all letters in $A_{z+1}A_{z+2}...A_{z+\ell}$ and $B_{z+1}B_{z+2...}B_{z+\ell}$ based on Fig. 7.

4.3) Calculate the *K_D* value between *A* and *B*:

$$K\_D = \sum_{i=z+1}^{z+\ell} \left| co_{A_i}, co_{B_i} \right| \qquad (13)$$

5) If $s_4 \geq s_i, i = 1,2,3$, then *A* is a transposition (Type 4) error.

5.1) Detect the transpose $A_{z+1}A_{z+2}$ in *A*,

5.2) Get the coordinates of $A_{z+1}A_{z+2}$ based on Fig. 7.

5.3) Calculate the *K_D* value between *A* and *B* using the following formula:

$$K\_D = \left| co_{A_{z+1}}, co_{A_{z+2}} \right| \qquad (14)$$

The *K_D* feature is useful when two candidates have the same distance to the misspelled word, but one is more likely to be a good correction if the error character distance on the keyboard is smaller than the other candidate. For instance, for the misspelled word “ans”, “and” is more likely to be a good correction than “ant” because “d” is adjacent to “s” on the QWERTY keyboard.

- **Probability of prefix 2-grams**: This is the probability of the appearance of the 2-gram made up of the correction candidate *C* and the preceding token *R* of the typo, which can be found in DB_5.
- **Probability of suffix 2-grams**: This is the probability of the appearance of the 2-gram made up of the correction candidate *C* and the subsequent token *S* of the typo, which can be found in DB_5.
- **Probability of centered 3-grams**: This is the probability of the appearance of 3-gram made up of the preceding token *R* of the typo, the correction candidate *C*, and the subsequent token *S* of the typo, which can be found in DB_5.

The *NN_TC_Conf* is trained with a set of misspelled words and their correction candidates. The misspelled words were detected in the training documents, and each candidate of each misspelled word is labeled with a confidence value. Specifically, the training data consist of tuples in the form of (x, $C(x)$, $conf(C(x))$), a typo x, its correction candidate $C(x)$, and $conf(C(x))$, the confidence about the candidate $C(x)$ being the correct word for typo x. For each pair of (x, $C(x)$), we generate the above 10 features as the input vector to the neural network, and use $conf(C(x))$ as the target confidence value of $C(x)$ being the correct word to replace typo x. After training, for any pair of typo x' and a correction candidate $C(x')$, the neural network *NN_TC_Conf* is used to generate a real value $conf(C(x')) \in [0,1]$ based on the ten features extracted from (x', $C(x')$), where $conf(C(x'))$ represents the confidence about $C(x')$ being the correct word to replace typo x'.

3.2 Intelligent Typo Detection and Correction (ITDC)

Fig. 14 gives an overview of the proposed automatic typo correction system, Intelligent Typo Detection and Correction (ITDC). The ITDC system contains four major computational components, typo detection and correction candidate generation, word boundary error correction, abbreviation processing, and correction candidate confidence generation and candidate selection. For a given text document, the ITDC system first detects typos and generates a candidate list using the general and domain specific knowledge about typos described in Section 2. It then detects and corrects two types of spelling errors: word boundary errors and uncommonly used abbreviations. Correction candidates for remaining non-word errors are then generated and weighted, and the best candidate is selected as the output from the ITDC system.

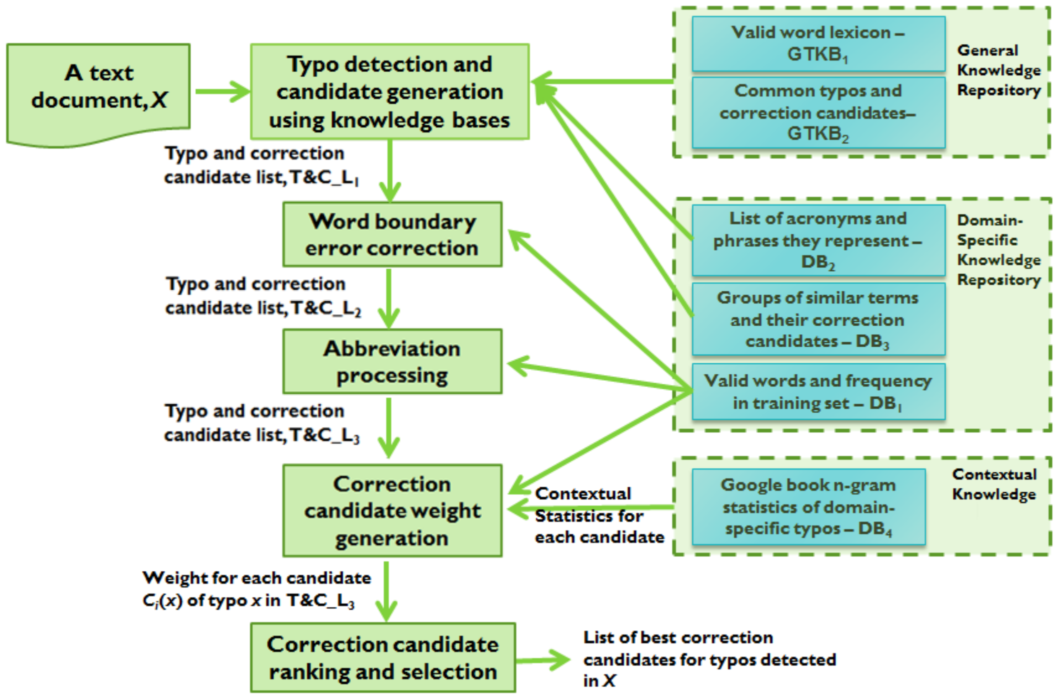

A text document, X
Typo detection and candidate generation using knowledge bases
Typo and correction candidate list, T&C_L1
Word boundary error correction
Typo and correction candidate list, T&C_L2
Abbreviation processing
Typo and correction candidate list, T&C_L3
Correction candidate weight generation
Weight for each candidate Ci(x) of typo x in T&C_L3
Correction candidate ranking and selection
List of best correction candidates for typos detected in X
Valid word lexicon – GTKB1
Common typos and correction candidates– GTKB2
General Knowledge Repository
List of acronyms and phrases they represent – DB2
Groups of similar terms and their correction candidates – DB3
Valid words and frequency in training set – DB1
Domain-Specific Knowledge Repository
Google book n-gram statistics of domain-specific typos – DB4
Contextual Knowledge
Contextual Statistics for each candidate

Fig. 14. Overview of ITDC (Intelligent Typo Detection and Correction) system

3.2.1 Typo detection and correction candidate generation

When a document is send to ITDC system for typo detection and correction, each term within the document is checked for its validity using the general knowledge base, $GTKB_1$, the lexicon. If a word is not in $GTKB_1$, then it is considered a typo. The ITDC system then generates a list of correction candidates based on $GTKB_2$, the general language typo list, DB_2, the list of acronyms commonly used in the application domain, and DB_3, the list of similar word groups and their correction candidates. The algorithm is described as follows.

Algorithm for typo detection and correction candidate generation

1) For each word *x* on the term list of the input document, check whether $x \in GTKB_1$. If it is, then exit, since *x* is not a typo.

2) If *x* is a commonly misspelled word found in $GTKB_2$, then use its correction candidate, *C*(*x*), provided by $GTKB_2$ as the correction candidate for *x*, add (*x*, *C*(*x*)) to $T\&C\_L_1$, and exit.

3) If there is a $(ac, ex(ac)) \in DB_2$ such that *x* = *ac*, then *C*(*x*) = *ex*(*ac*), add (*x*, *C*(*x*)) to $T\&C\_L_1$, and exit.

4) If there is a $(g, g\_w) \in DB_3$ such that $x \in g$, add (*x*, *g_w*) to $T\&C\_L_1$, and exit.

5) Search in $GTKB_1$ to find the *N* valid words that best match with *x* using the Levenshtein distance function, which are denoted as $C_i(x), i = 1,2,...N$, add $(x, \{C_1(x), C_2(x),...C_N(x)\})$ to $T\&C\_L_1$, and exit.

The Output from the above algorithm is $T\&C\_L_1$, which contains all the typos found in an input document along with up to *N* correction candidates for each typo. Examples of entries in the list $T\&C\_L_1$ are shown in Table 3.

Table 3 EXAMPLE OF ENTRIES IN T&C_L_1

Misspelling	Correction candidate List
abandonned	abandoned (Generated from GTKB2, in Step 2 above)
KOEC	Key On Engine Cranking (Generated from DB2, in Step 3 above)
engien	Engine (Generated from DB3, in Step 4 above)
ststem	system, systems, state, states, stem (Generated from GTKB1, in Step 5 above)
noisehard	noseband (Generated from GTKB1, in Step 5 above, not a good candidate)
diagn	diag, drag, drain, drags, dragon (Generated from GTKB1, in Step 5 above, not a good candidate list)

3.2.2 Word boundary error detection and correction

Word boundary errors are defined as words either missing white space characters between multiple words (run-on error), or a valid word split by a white space (split error) [16]. It is very important to process word boundary errors separately, since they are very different from the other typos. For instance, for the split error "or dered", the correction candidate list could only be generated on typo "dered", because "or" is a valid word. The following describes the two algorithms we developed to solve word boundary errors, one for split error correction, and another for run-on error correction.

Split Error Correction algorithm

1) For each typo x in T&C_L_1, find its adjacent words in the input document X, w_1 and w_2.

2) Check the dictionary for $wx = w_1 \| x$, and $xw = x \| w_2$.

If *wx* is a valid word, calculate the probability of appearances of w_1, *x* and *wx*, denoted as $p(w_1)$, $p(x)$ and $p(wx)$, where $p(w_1) = \frac{f(w_1)}{f(*)}$, $f(w_1)$ denotes the frequency of word w_1 appearing in the knowledge base DB_1, and $f(*)$ denotes frequencies of all words in DB_1. $p(x)$ and $p(wx)$ are calculated using the same way.

3) If w_1 is a valid word and $p(wx) > p(x)$, update T&C_L1 by setting *wx* as the only correction candidate of two terms "w_1 x" and exit.

4) If w_1 is also typo and $p(wx) > p(x)$, $p(wx) > p(w_1)$, update T&C_L1 by setting *wx* as the only correction candidate for the two terms "w_1 x" and exit.

5) Step 6: Repeat Step 3 to process *xw*.

We use one example to illustrate the rationale behind Step 4 and 5. Let w_1= "or", x = "dered". We have wx = "ordered". If p("ordered") > p("dered"), which implies that only if the combined word *wx* appears more often than the typo *x* in the training documents, then we use *wx* as correction for "w_1 x". If both w_1 and *x* are invalid, Step 5 makes sure that *wx* has to appear more often than both w_1 and *x*. Because if "*wx*" never appeared in *D*, but "*x*" appeared many times, then *x* might have its own correction candidate instead of "*wx*".)

Run-on Error Correction algorithm

To detect and correct run-on typos, the system checks each typo, *x*, on the list T&C_L1 to see if it can be split into two valid words w_1 and w_2, while $p(w_1) > 0, p(w_2) > 0$. If so, let (w_1 w_2) be the only correction candidate for typo *x*, and update T&C_L1 accordingly. If *x* can be separated into multiple pairs of valid words, for instance, both $x = w_1 \| w_2$ and $x = w_3 \| w_4$, we use Bayes formula to calculate their occurrence probability in training data *D*:

$$P(w_1w_2) = P(w_1 \mid w_2)P(w_2) = \frac{f(w_1w_2)}{f(*w_2)} \times \frac{f(w_2)}{f(*)},$$
$$P(w_3w_4) = P(w_3 \mid w_4)P(w_4) = \frac{f(w_3w_4)}{f(*w_4)} \times \frac{f(w_4)}{f(*)} \qquad (15)$$

Where $f(*w_2)$ denotes the frequency of any two word sequence that has w_2 as the second word. The pair of words with higher appearance probability is selected as the candidate to replace typo x.

The output of these two algorithms is an updated typo list of T&C_L$_1$, denoted as T&C_L$_2$, in which the word boundary typos are corrected. Table 4 shows the results after applying the two word boundary algorithms to the T&C_L$_1$ shown in Table 3.

Table 4 EXAMPLE OF ENTRIES IN T&C_L$_2$

Misspelling	**Correction candidate List**
abandonned	abandoned
KOEC	Key On Engine Cranking
engien	Engine
ststem	system, systems, state, states, stem
noisehard	noise hard **(updated after word boundary error correction)**
diagn	diag, drag, drain, drags, dragon

3.2.3 Abbreviation processing

Abbreviations are defined as the segment of a valid word starting with the first character of the valid word [4]. For instance, “diagn” is the abbreviation of “diagnose”. For these abbreviations, simple similarity comparison methods may not find the correct candidates. For example, a candidate list for "diagn" generated based on word distances may contain "diag, drag, drain, drags, dragon", but not correct candidate.

In many note taking or records keeping applications such as vehicle diagnostic records, people often type in the first several characters of a word as its abbreviation. For example, "conn" represents "connection", "comm" represents "communication", and etc. In order to detect those uncommon abbreviations, we compare each typo, x, in *D_typos* with every valid word, w, in DB_1. If x matches the beginning $|x|$ letters in w, add w to *x's* candidate list, where $|x|$ denotes the number of letters in x.

The output of this stage is a further updated typo list $T\&C\_L_3$ where the abbreviations are processed. Table 5 shows the result of this process on the words shown in Table 4.

Table 5 EXAMPLE OF ENTRIES IN $T\&C\_L_3$

Misspelling	**Correction candidate List**
abandonned	abandoned
KOEC	Key On Engine Cranking
engien	Engine
ststem	system, systems, state, states, stem
noisehard	noise hard
diagn	diagnose, diag, drag, drain, drags, dragon **(updated after abbreviation processing)**

3.2.4 Correction candidate weight generation, ranking and selection

In an automatic typo correction system, when more than one correction candidates are generated, a crucial task is to determine which word within the correction candidate list should be selected to replace the misspelled word. Our solution to this problem is to use contextual knowledge to re-rank the candidates. The following algorithm is developed to rank the candidates of typo *x* based on the n-gram statistics knowledge contained in DB_4 and the neural network NN_TC_Conf.

Algorithm for ranking typo candidates based on contextual knowledge

For $n = 3 \sim G$, where *G* denotes the maximum length of n-gram we used in generating DB_4:

1) For a typo *x* detected in the testing document *X*, extract the prefix n-grams, suffix n-grams and centered n-grams of *x* from *X*, which are denoted as $ngram_{pre}(x) = R'_{n-1}$, $ngram_{suf}(x) = S'_{n-1}$, and $ngram_{cen}(x) = \{R'_m, S'_m\}$. This is similar to the process of generating n-grams for typos in training documents discussed in section 2.3.

2) For each candidate word *C(x)* generated for *x*, let $H_n(C(x)) = 0$, where $H_n(C(x))$ represents the normalized frequency of *C*(*x*) appears in the Google book n-gram corpus.

2.1) If $R'_{n-1} \| C(x)$ equals to any ng_i in DB_4, where $i = 1,2,...L$, $H_n(C(x)) = H(ng_i)$.

2.2) If $C(x) \| S'_{n-1}$ equals to any ng_i in DB_4, where $i = 1,2,...L$, $H_n(C(x)) = H_n(C(x)) + H(ng_i)$.

2.3) If $R'_m \| C(x) \| S'_m$ equals to any ng_i in DB_4, where $i = 1,2,...L$, $H_n(C(x)) = H_n(C(x)) + H(ng_i)$.

The output from this process is a weighted frequency $F_x(C(x))$ for each *C*(*x*) generated for typo *x*, where $F_x(C(x)) = \sum_{n=3}^{G} H_n(C(x))$. Take the same example typo "whele" as we discussed in section 2.3, two candidates "where" and "wheel" generated for "whele" are found in phrases of DB_4. Suppose $G = 3$, we have $F_{whele}(where)$=1.63E-09, and $F_{whele}(wheel)$=2.68E-07, which indicates that the "wheel" is better than "where" as the correction candidate for typo "whele."

Let $\{C_i(x), i = 1,2,...N\}$ be N correction candidates for each typo x in T&C_L$_3$.

- For each $C_i(x)$ generated for x, within in the time period yr_1 to yr_Y, calculate weighted frequency $F_x(C(x))$ in google n-gram corpus.
- For each pair of (x, $C_i(x)$), extract the 10 features discussed in section 2.4 and apply them to NN_TC_Conf, which outputs a confidence value $conf(C_i(x))$.
- Calculate the weight $V(C_i(x))$ for each $C_i(x)$, where $V(C_i(x)) = conf(C_i(x)) * F_x(C_i(x))$.
- Re-rank these N correction candidates $C_i(x)$ in descending order based on $V(C_i(x))$.
- For all $C_j(x)$ having the same value of $V(C_j(x))$, calculate $Dist((x, C_i(x)))$ as discussed above in (2) of section 2.2, and re-rank these correction candidates $C_j(x)$ for typo x based on this similarity distance.
- Select top N' correction candidates from re-ranked candidate list, as $\{C_i'(x), i = 1,2,...N'\}$.
- After correction candidate weight generation, ranking and selection, for each misspelled word x, the auto correction is quite straightforward: Check whether the first candidate $C_1'(x)$ in the re-ranked candidate list has $conf(C_1'(x)) > 0.5$. If so, replace the misspelled word x in text using $C_1'(x)$. Otherwise, x is flagged but not corrected, and top N' correction candidates are stored for manual correction afterwards.

The flowchart of candidate ranking and selection algorithm for each typo x in the testing document X is summarized in Fig. 15.

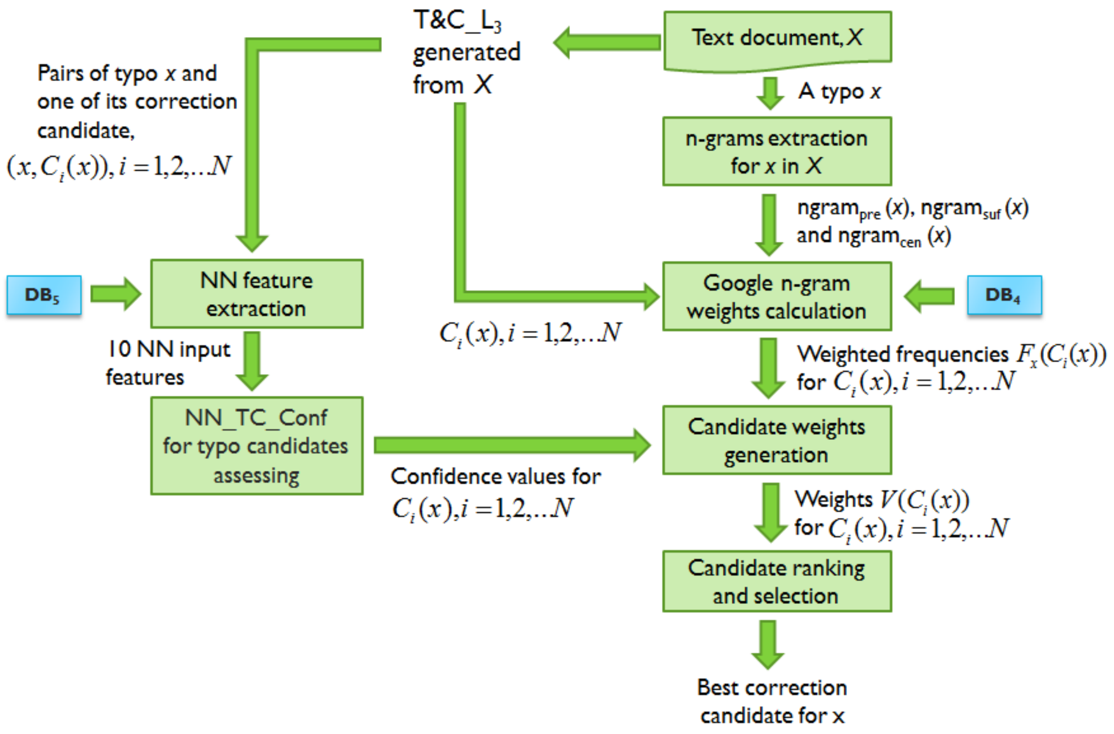

T&C_L$_3$ generated from X
Text document, X
Pairs of typo x and one of its correction candidate, $(x, C_i(x)), i = 1,2,\ldots N$
A typo x
n-grams extraction for x in X
ngram$_{pre}$ (x), ngram$_{suf}$ (x) and ngram$_{cen}$ (x)
DB$_5$
NN feature extraction
$C_i(x), i = 1,2,\ldots N$
Google n-gram weights calculation
DB$_4$
10 NN input features
Weighted frequencies $F_x(C_i(x))$ for $C_i(x), i = 1,2,\ldots N$
NN_TC_Conf for typo candidates assessing
Candidate weights generation
Confidence values for $C_i(x), i = 1,2,\ldots N$
Weights $V(C_i(x))$ for $C_i(x), i = 1,2,\ldots N$
Candidate ranking and selection
Best correction candidate for x

Fig. 15. Candidate ranking and selection

3.3 Empirical study

We conducted an empirical study in the application domain of automotive fault diagnostics text data mining, where the text documents are warranty claims and repairing descriptions recorded in verbatim. These text records contain rich information about the various cases of vehicle malfunctions, root causes, and repair processes, yet are difficult to manually extract knowledge or formulate reliable rules to associate an effective repair procedure with a given problem description. Moreover, the records have poor grammar structure, and contain many typos, self-invented abbreviations and domain specific terminologies, which are big challenges to data mining systems. We implemented the proposed ITDC system and applied it to the text documents in this application domain. The following subsections describe the constructed knowledge bases and experiment results.

3.3.1 Building general knowledge bases

In this study, we constructed the valid lexicon, $GTKB_1$, from the WinEdt English_US and English_UK dictionary developed by Patrick Daly [20]. WinEdt is a powerful and versatile text editor for Windows, with a strong predisposition towards the creation of LaTeX documents. $GTKB_1$ contains more than 150,000 valid English words, and is used to recognize non-word errors. A list of common typos collected by Wikipedia Typo Team was used as our general knowledge base $GTKB_2$. $GTKB_2$ contains 4238 misspellings frequently appearing in online documents throughout Wikipedia.

3.3.2 Building domain specific knowledge bases

In this empirical study, we used a set of 200,000 customers' claim reports on various vehicle problems provided by an automotive company as the training document set *D* to generate domain specific knowledge bases DB_1, DB_2 and DB_3. Initially 10233 distinct index term words were extracted from *D*, and stored in DB_1 along with their term frequencies. By dictionary searching, we found 1763 distinct typos with length less than *l_ac* = 5, and, among them, 144 acronyms were finally identified and extracted. Examples of nontrivial acronyms in the DB_2 knowledge base are: CEL - Check Engine Light, CKT - Circuit, DLC - Data Link Connector, KOEC is used frequently in the context of DTC (Diagnostic Trouble Code) extraction referring to "Key On Engine Cranking", etc. We also applied word grouping algorithm presented in section 2.2 to obtain 3972 word groups, which are used as DB_3.

In the experiments, we used Google Book corpus: American English, which contains words that occurred in books published during 1980 ~ 2008 to generate the contextual knowledge base DB_4, and, due to the space limitation of storing the dataset, only the 3-gram phrases of each typo were extracted. The training data *D* was used to generate domain specific contextual knowledge base DB_5.

3.3.3 Typo detection and correction

Two sets of test documents, T_1 and T_2, provided by two different automotive manufacturers were used to evaluate the ITDC system. The first testing set T_1 contains 603 freeform technician verbatim problem descriptions. The second testing set T_2 contains 580,000 vehicle fault diagnostic records. We set the thresholds for word length and average word grouping distance discussed in section 2.2 as follows: $\tau_l = 2$, where only words with length larger than 2 will be processed, and in approximate string similarity matching, $\tau_\mu = 0.3$, which is the average distance threshold for word comparing.

ITDC used GTKB$_1$ to detect typos in T_1 and T_2. There were 392 typos detected in T_1 and 29113 typos detected in T_2. From those typos, the word boundary processing algorithms detected and corrected 74 split errors and 59 run-on errors in T_1, and 1292 split errors and 5273 run-on errors in T_2. The false alarm in T_1 and T_2 was 0% and 0.4%, respectively. Note here, most of the wrong corrections in T_2 were those typos being detected incorrectly as run-on errors. For examples, "holtline", which should have been detected and corrected as "hotline," was detected as a run-on error, so it was split into "holt line"; "performace", which should have been corrected to "performance," was detected incorrectly as a run-on error, and so it was split into "perform ace."

The abbreviation process corrected 14 non-word errors in T_1, and 242 non-word errors in T_2. Examples of detected word boundary errors and abbreviations are shown in Table 6 and Table 7. These terms are domain specific, so the domain specific knowledge played an important role in correcting these typos. Particularly, these abbreviations are neither commonly used in news articles nor in general web documents.

Table 6 EXAMPLE OF WORD BOUNDARY ERRORS

Split error	**Correction**
thro ttle	throttle
repla ced	replaced
shie ld	shield
hea vily	heavily
exc essive	excessive
Run-on error	**Correction**
sensorand	sensor and
connectionat	connection at
roadtest	road test
differentialpossible	differential possible
drivebelt	drive belt

Table 7 EXAMPLE OF UNCOMMON ABBREVIATIONS

Abbreviations	**Correction**
conn	connection
diagn	diagnose
cont	continue
comm	communication
veh	vehicle
diff	different
eng	engine
cust	customer
sig	signal

The knowledge bases $GTKB_2$ (general misspellings), DB_2 (Domain-specific acronyms) and DB_3 (Domain-specific word groups) were used to correct 29 non-word errors in T_1, and 1127 non-word errors in T_2. Some examples are listed in Table 8 - 10.

Table 8 EXAMPLE OF TYPOS RECOGNIZED BY GTKB$_2$

Error	**CORRECTION**
recieved	received
continous	continuous
intergration	integration
ocasionally	occasionally
bewteen	between
procede	proceed
fromed	formed
neccesary	necessary
taht	that
thsi	this
reponse	response

Table 9 EXAMPLE OF TYPOS RECOGNIZED BY DB$_2$

Error	**CORRECTION**
ECU	Engine Control Unit
EVAP	Evaporative Emission
IPC	Instrument Panel Cluster
KOEC	Key On Engine Cranking
TP	Torque Converter Clutch
VSS	Vehicle Speed Sensor
TCC	Torque Converter Clutch

Table 10 TYPOS RECOGNIZED BY DB_3

Error	**CORRECTION**
erractic	erratic
trac	track
whl	wheel
programed	programmed
manuvers	maneuver
chekc	check
fse	fuse
exhuast	exhaust
sytem	system
tryed	tried

The training data for the neural network, NN_TC_Conf are processed as follows. First we collected a set of 2230 freeform technician verbatim documents from the training data set *D*, which contained 1547 typos. We used Levenshtein distance algorithm to generate up to five candidates for each misspelled word by finding the closest valid words in DB_1 to the typo. These typos and their respective candidates are manually labeled as high or low confidence for being the correct candidates. This process generated 783 likely candidates and 6166 unlikely candidates.

As discussed in section 3.4, for each typo detected, we set $G = 3$, where *G* denotes the maximum length of n-gram we look into, and generated 20 correction candidates first, i.e. $N = 20$. After the candidate weight generation and ranking, we selected top 5 correction candidates, where $N' = 5$. Note *G*, *N* and *N'* were used in section 3.4.

For the purpose of comparison, we applied the two state-of-art spell checkers: Google Spell check and Aspell check, to the same two testing document sets, T_1 and T_2. We use the following three Suggestion Intelligence First (SIF) measures [6,7] to evaluate the performances of the typo correction systems:

$$\text{SIF} = \frac{\text{Total correct suggestions found first on list}}{\text{Total number of typos}}$$

$$\text{SIF3} = \frac{\text{Total correct suggestions found in top 3 of list}}{\text{Total number of typos}}$$

$$\text{SIF5} = \frac{\text{Total correct suggestions found in top 5 of list}}{\text{Total number of typos}}$$

The performance results of the three systems are shown in Table 11 and Table 12. The proposed ITDC system made 3.83% false auto correction on T_1, and 3.54% false auto correction on T_2.

Table 11 Performance Comparison with State-of-art Spell Checkers on T1

	SIF5	**SIF3**	**SIF**
Google Spell	50.77%	48.72%	45.41%
Aspell	27.81%	26.02%	25.51%
ITDC without candidate ranking process	54.85%	52.81%	50.77%
ITDC after candidate ranking process	65.56%	64.28%	62.24%

Table 12 Performance comparison with state-of-art spell checkers on T2

	SIF5	**SIF3**	**SIF**
Google Spell	57.96%	55.05%	51.29%
Aspell	44.25%	42.20%	39.51%
ITDC without candidate ranking process	63.87%	61.61%	60.47%
ITDC after candidate ranking process	68.43%	67.34%	65.10%

Table 13 shows a few examples of typo corrections made by the ITDC system, the Google and Aspell systems. It is obvious that ITDC system outperforms these two systems by abbreviation processing, word boundary errors and correction candidate re-ranking based on neural learning of the confidence of candidate correctness and n-gram statistical analysis.

Table 13 Correction candidate List comparison

Typo		**Correction Candidates**				
nece	**Google**	neck	Becca	Mecca	mecca	enc
	Aspell	neck				
	ITDC	necessary				
whele	**Google**	whale	wheel	while	whole	Wheeler
	Aspell	where	wheel	whale	whelp	while
	ITDC	wheel	while	where	whole	whelp
speend	**Google**	spend	speed	spent	spawned	spurned
	Aspell	spend	speed			
	ITDC	speed	spread	spend	spent	steed
servi	**Google**	servo	serve	server	serf	servos
	Aspell	serve				
	ITDC	service	serve	servo	swerve	

The accuracy of all three spell checker systems evaluated above may not be considered very high. This is due to the fact that the testing set contains many non-word typos or domain-specific abbreviations that cannot be recognized easily, such as “bjb”, “btwn”, “ops”, “ssms”, etc. These errors were detected and marked as typos by the ITDC system, but left without correction.

In order to prove the assumption that typo correction could provide a better quality and more comprehensible text for both human and machines and help with further text processing tasks, we conducted text categorization on the same text collections, T_1 and T_2, as those used in the above typo correction experiments, in which each document has a category label. For both of the dataset, we use conventional VSM model discussed in section 2.4.1 as text representation approach, with the same local and global weighting scheme, tf-idf approach [67]. From each dataset, we choose 2/3 documents from each class as training set, and the remaining 1/3 documents as testing set, and conduct 3-fold cross validation to get the average accuracy of the system. More details about text categorization could also be found in the following Chapter 4. The accuracy of the text categorization is measured using the following evaluation metric:

$$\text{Accuracy} = \frac{\text{Total number of testing documents correctly classified}}{\text{Total number of testing documents}}$$

The experiment results are shown in the following Fig. 16 and Fig. 17, from which it is obvious that typo correction improves text categorization accuracy by 2% and 5%, and reduce term feature space by 7.7% and 14%, respectively. We can observe that this improvement is more significant especially in larger dataset where typos are more frequently occurs, e.g., text collection T_2, since it greatly reduces the noisy terms and merges topographically similar terms.

In terms of efficiency of ITDC, all the knowledge bases generated above are automated programs that extract and store general language knowledge and domain specific knowledge from digital resources effectively. The only manual work involved are to determine which resources to use and preparing the data, as well as labeling training data for NN_TC_Conf. These steps usually take only 3-4 hours before typo detection and correction.

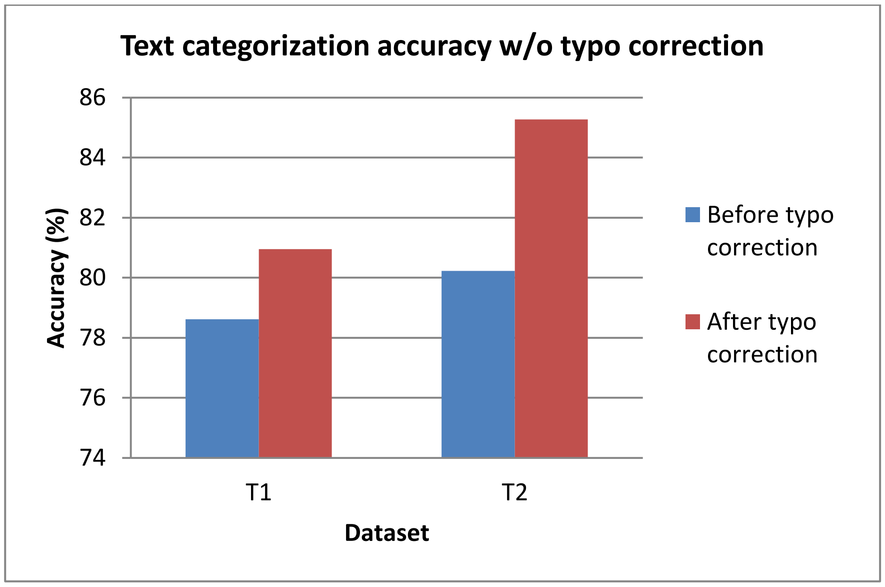


Fig. 16. Example of text categorization accuracy w/o typo correction

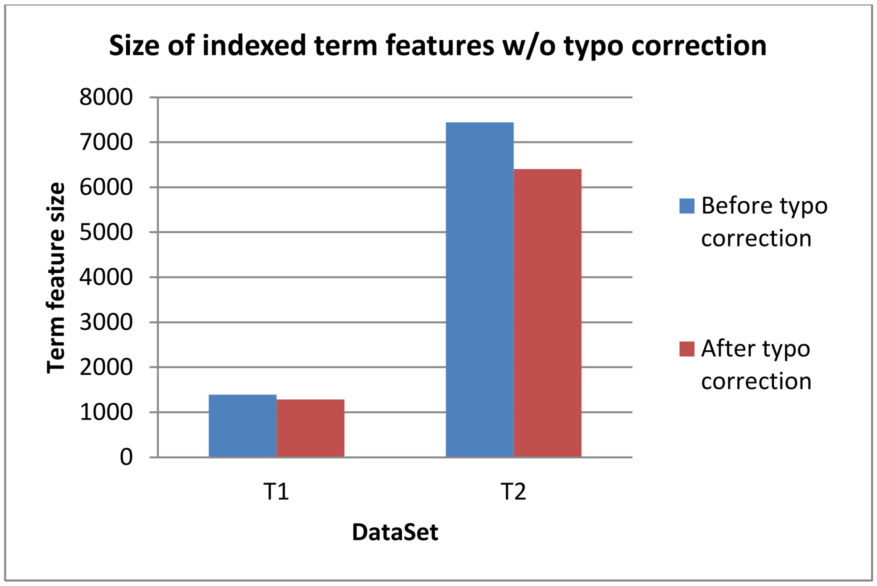


Fig. 17. Example of text categorization feature size w/o typo correction

CHAPTER 4. TEXT CATEGORIZATION BASED ON MACHINE LEARNING, STATISTICAL MODELING AND ONTOLOGY NETWORK

As mentioned in section 2.5.1, the traditional VSM model does have its strength: it is efficient and provides a compact way of text representation, instead of fully understanding the content using natural language processing techniques, which is usually time and space consuming. However, since only single word information is considered, text categorization accuracy may be affected if single words do not fully interpret the document content, meaning information such as word co-occurrence, word context within documents and semantic ambiguity of words including synonymy and polysemy, are missing. Therefore, a text representation model that has sophisticated structure and as inclusive as possible in terms of text information is of great necessity. In this chapter, we present our research work in text categorization by introducing VSM-based text representation models using machine learning, statistical modeling and ontology networks. We use an innovative hybrid text mining framework, which contains a global weighting scheme, a VSM model built from WordNet ontology network, and a VSM model augmented with statistical topic modeling. Fig. 18 illustrates the proposed framework. Our system takes in the training document collection and generates a list of indexed terms. After that, each indexed terms are weighted, and the document corpus is modeled by traditional VSM as a weighted TD matrix. PLSA model is applied to generate a "latent topic" level (LTD) matrix, and WordNet ontology is feed into the system to generate a new term-document matrix, and a "concept" level (CD) matrix. These matrixes are then combined together for final document representation, and used for SVM classifier training. More details will be further discussed in the following sections.

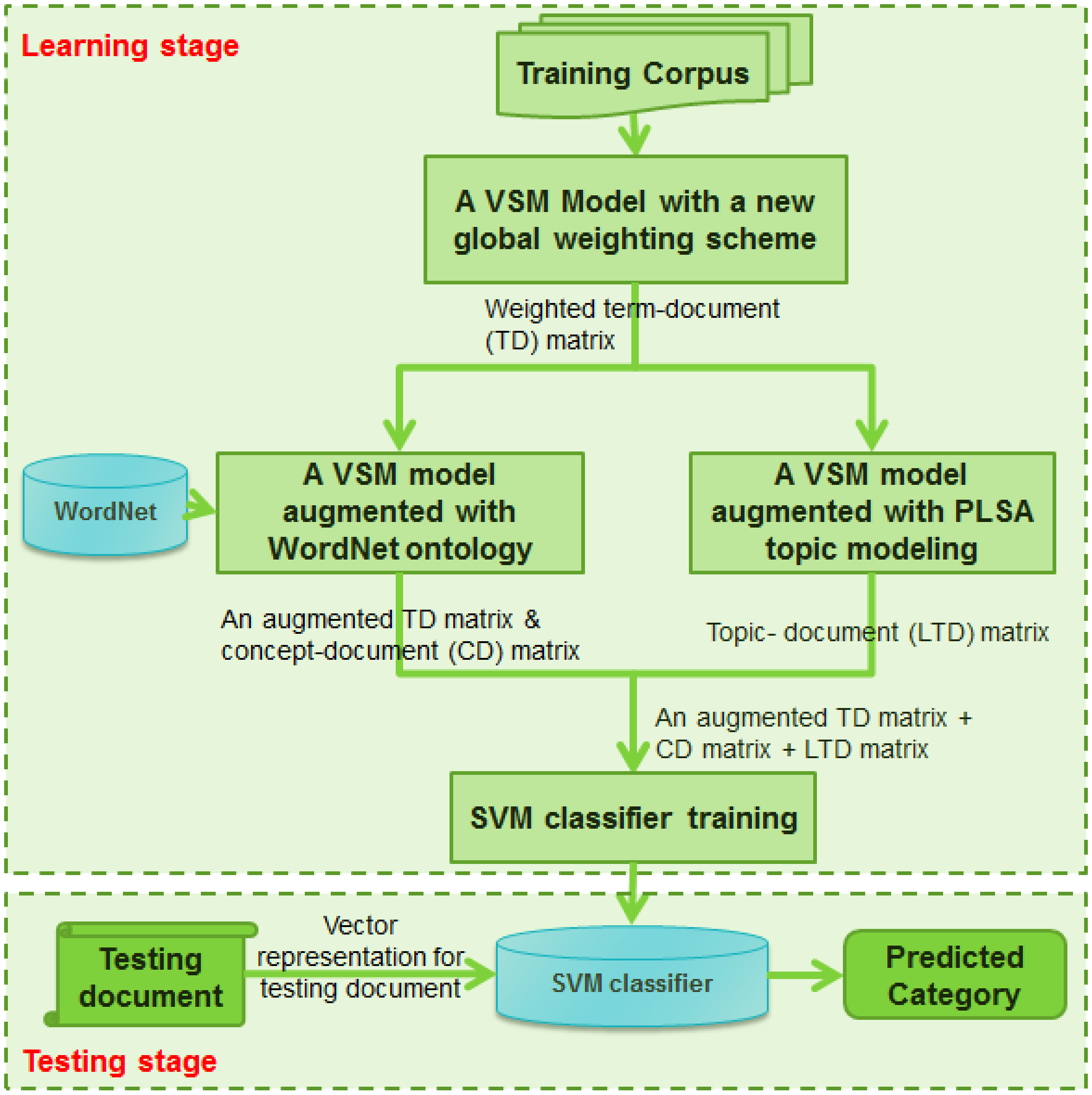

Learning stage
Training Corpus
A VSM Model with a new global weighting scheme
Weighted term-document (TD) matrix
WordNet
A VSM model augmented with WordNet ontology
A VSM model augmented with PLSA topic modeling
An augmented TD matrix & concept-document (CD) matrix
Topic- document (LTD) matrix
An augmented TD matrix + CD matrix + LTD matrix
SVM classifier training
Testing document
Vector representation for testing document
SVM classifier
Predicted Category
Testing stage

Figure 18. Proposed text categorization model framework

4.1 Text categorization based on VSM and PLSA topic modeling

This section discusses our work in building a VSM model using an entropy-based global weighting scheme, using PLSA topic modeling to extract latent topics and build a VSM model that reflect word relationships, as well as using semi-supervised PLSA topic modeling to build a VSM model that reflect document relationships, based on pre-defined document "connectivity" information.

4.1.1 A VSM Model with a new global weighting scheme

As discussed in Section 2.4.1, text document is usually represented by VSM for the ease of computation and analysis. A vector space model should be built based on carefully selected terms and weighting schemes. More specifically, for a given set of training documents *Tr*, $Tr = \{D_1, D_2, ..., D_N\} = T_1 \cup T_2 ... \cup T_C$, where D_l is the l^{th} training document, *C* is the number of document categories, and T_c is the set of documents that belong to category *c*, $c = 1,2,...,C$, our vector space model is built through the following machine learning process:

First of all, we generate an indexed term list from *Tr*, denoted as *T_L*, $T\_L = \{t_1, t_2, ..., t_K\}$, where t_i is the i^{th} indexed term, through a number of preprocessing tasks, including word tokenization, symbol and punctuation removing, automatic typo correction as discussed in Chapter 3, stopping word removal and low-frequency term removal, etc. While generating the list of indexed term words, we need to keep only content bearing words, implying that the function words having both low and high frequency have to be removed [68]. As a result, we removed the high frequency stop words at the first stage, and then set up a term frequency threshold τ as a filter for low frequency words.

Secondly, for a document $D_l \in Tr$, its vector representation is defines as following:

$$W_{D_l} = [w_{t_1,D_l}, w_{t_2,D_l}, \ldots, w_{t_K,D_l}],$$
$$w_{t_i,D_l} = tf_{il} * GW_i, i = 1,2,\ldots,K.$$

Where tf_{il} is the occurrence frequency of term t_i within D_l, GW_i is a global weight for term t_i, and K is the number of term features.

VSM models based on appropriate term weighting schemes is particularly essential for information retrieval and text categorization [67]. An appropriate global weighting scheme should be applied to each indexed term with the purpose of reducing or enhancing the effect they have on particular documents. As mentioned in section 2.4.1, although there are plenty of global weight approaches available, most of them are designed for the entire dataset, i.e., in *idf* global weighting, $idf = \log_2\left[\frac{ndocs}{df_i}\right] + 1$, df_i denotes the total number of documents in the document collection that contain term t_i, and *ndocs* represents total number of documents in the whole document collection *Tr*. However, based on our observation, important term words or their synonyms often appear frequently in documents within a specific category, especially when the user defined categories are highly relevant to some specific keywords [24]. As a result, we developed the following category-entropy global weighting scheme, denoted as *CE_W*:

- For each term t_i in the term list *T_L*, calculate the proportion of the documents in *Tr* that contain t_i within *C* different categories.

  $$p_{ij} = \frac{N\_c_{ij}}{N\_c_j}, j = 1,2,\ldots C,$$

where N_c_{ij} is the number of documents within the j^{th} categories that contains t_i, and N_c_j is the total number of documents in the j^{th} category.

- Normalize p_{ij}, so that $\alpha_{ij} = \frac{p_{ij}}{\sum_{j=1}^{C} p_{ij}}$.
- Calculate the entropy with respect to t_i: $E_i = \sum_{j=1}^{C} -\alpha_{ij} \log \alpha_{ij}$.

  The entropy measure is a good indicator of how term t_i is distributed over different document categories. The higher the entropy, the less important item t_i is, since it is more evenly distributed among different document categories.
- Calculate the global weight *CE_W* for t_i: $CE\_W_i = 1 - \frac{E_i}{\log C}$, where *C* is the total number of categories. This global weight function gives more weights to terms that have small entropy values.

We will show in Chapter 5 that the category-entropy based global weight function performs much better than the most widely used inverse document frequency (*idf*) method.

At the end of VSM generation step, the output is a TD matrix $M_0, M_0 = [W_{D_1}^T, W_{D_2}^T, ..., W_{D_N}^T]$. For a previously unseen document D_u, we generate its vector representation W_{D_u} in the same manner, $W_{D_u} = [w_{t_1,D_u}, w_{t_2,D_u}, ..., w_{t_K,D_u}]$, for the testing purpose.

4.1.2 A VSM augmented with PLSA topic modeling

In this section, we mainly discuss the details of how to use PLSA topic modeling introduced in section 2.4.3 to extract latent semantic topics from text, which are used to generate an augmented VSM model for text representation and help improving the accuracy of text categorization tasks.

4.1.2.1 Learning PLSA model from training documents

As already presented in section 2.4.3, PLSA model is a well-known statistical language model mostly used for unsupervised text clustering and information retrieval. The starting point of PLSA is the term-document frequency (TDF) matrix before applying global weight scheme, and it follows the bag-of-words assumption, in which each word appears independently, and the occurring order of each word is not considered. As shown in Fig. 6, $P(D)$, $P(z \mid D)$, $P(t \mid z)$ represents the probabilities of observing a document *D*, a latent topic *z* occurring in *D*, and word *t* belonging to *z*, respectively. The generative process of each document-word pair in the text corpus, *Tr*, is shown as following:

1. Select a document *D* from *Tr* based on $P(D)$.
2. Pick a topic *z* according to $P(z \mid D)$.
3. Given *z*, generate a word *t* based on $P(t \mid z)$.

The hidden variable set during this process, denoted as θ, $\theta = \left(P(z \mid D), P(t \mid z)\right)$, is what we are interested in and want to estimate, for each word-topic pair and topic-document pair.

Again, we know that the joint probability of each document-word pair could be derived as following, based on Bayes' Theorem [40]:

$$P(D,t) = P(D)P(t \mid D) = P(D)\sum_{z}(P(t \mid z)P(z \mid D)),$$

Here, $P(t \mid D)$ is derived based on Bayes' rule, with the following two assumptions [74]: First, observation of document-word pairs (D,t) are assumed to be generated independently, which is corresponding to the "bag-of-words" approach. Secondly, given latent topic *z*, *t* is also generated independently of *D*. Therefore, we have:

$$P(t \mid D) = \frac{P(D,t)}{P(D)} = \frac{\sum_z P(D,t,z)}{P(D)} = \frac{\sum_z (P(D,t \mid z)P(z))}{P(D)} = \frac{\sum_z (P(D \mid z)P(t \mid z)P(z))}{P(D)}$$
$$= \frac{\sum_z (P(z \mid D)P(t \mid z)P(D))}{P(D)} = \sum_z (P(t \mid z)P(z \mid D))$$

The likelihood function of the entire document collection, *Tr*, could also be derived as: $P_L = \prod_D \prod_t P(D,t)^{n(D,t)}$, based on the observation of all document-word pairs, where $n(D,t)$ denotes the frequency of word *t* appears in document *D*. Our objective is thus estimating the hidden variables by maximizing this likelihood function. Because it is difficult to maximizing the above exponential likelihood function, it is more convenient to work with its logarithm, called the log-likelihood. The objective of this estimation is thus to maximize the log–likelihood function of θ, as shown in formula (1):

$$L(\theta) = \sum_{D,t} n(D,t) \log(P(D) \sum_z P(t \mid z)P(z \mid D)), \text{ (1)}$$

Since $P(D)$ is not related to the parameter we want to estimate and we assume that it is constant among documents in *Tr*, and also we assume that for each document, $P(D)$ is a constant value. As a result, we then have:

$$\arg\max(L(\theta)) \propto \arg\max \sum_{D,t} n(D,t) \log(\sum_z P(t \mid z)P(z \mid D)) \quad (2)$$

As mentioned in section 2.4.3, (2) can be solved using Expectation Maximization (EM) algorithm [75]. We now take a deep look at the derivation of why and how EM algorithm can be applied for estimating θ.

Derivation of EM algorithm

From (2), we can see that it is still difficult to find the solution for θ with the "$\log\Sigma$" format. As a result, we introduce a distribution over topic z into (2), denoted as $A(z)$, where $A(z) \geq 0$ and $\sum_z A(z) = 1$. We then have:

$$(2) = \arg\max \sum_{D,t} n(D,t) \log(\sum_z A(z) \frac{P(t \mid z)P(z \mid D)}{A(z)}), \quad (3)$$

Based on the law of unconscious statistician [106], let $g(z) = \frac{P(t \mid z)P(z \mid D)}{A(z)}$, we have $E(g(z)) = \sum_z g(z)A(z)$. Therefore,

$$(3) = \arg\max \sum_{D,t} n(D,t) \log E(\frac{P(t \mid z)P(z \mid D)}{A(z)}) \quad (4)$$

From (4), it is still difficult to find the solution for θ with "$\log E(g(z))$" format. However, because $\log E(\frac{P(t \mid z)P(z \mid D)}{A(z)})$ is a concave function, based on Jensen's inequality [110], we could find a lower bound function for (4):

$$\begin{aligned}(4) &\geq \arg\max \sum_{D,t} n(D,t) E(\log \frac{P(t \mid z)P(z \mid D)}{A(z)}) \\ &= \arg\max \sum_{D,t} n(D,t) \sum_z A(z) \log \frac{P(t \mid z)P(z \mid D)}{A(z)}\end{aligned}$$

Note here, when $\frac{P(t \mid z)P(z \mid D)}{A(z)} = \mu$, where μ is a constant, we have:

$$(4) = \arg\max \sum_{D,t} n(D,t) \sum_{z} A_r(z) \log \frac{P(t \mid z)P(z \mid D)}{A(z)} \qquad (5)$$

Because $\sum_{z} A(z) = 1$, $\sum_{z} P(t \mid z)P(z \mid D) = \mu$. Also, based on Bayes' rule and the independency of (D,t), we have:

$$P(z \mid D,t) = \frac{P(z,D,t)}{P(D,t)} = \frac{P(D,t \mid z)P(z)}{P(D)\sum_{z}(P(t \mid z)P(z \mid D))} = \frac{P(D \mid z)P(t \mid z)P(z)}{P(D)\sum_{z}(P(t \mid z)P(z \mid D))}$$
$$= \frac{P(z \mid D)P(t \mid z)P(D)}{P(D)\sum_{z}(P(t \mid z)P(z \mid D))} = \frac{P(z \mid D)P(t \mid z)}{\sum_{z}(P(t \mid z)P(z \mid D))}$$

Thus, $A(z) = \frac{P(t \mid z)P(z \mid D)}{\sum_{z} P(t \mid z)P(z \mid D)} = P(z \mid D,t;\theta_r)$, which shows that to find the maximum solution for (4), at every iteration *r*, $A(z)$ should be the posterior probability of *z*, with observed document-word pair (*D*, *t*). Therefore, EM algorithm could be used as following:

Computational steps of EM algorithm

Each iteration of EM algorithm consists of expectation step (E-step) and maximization step (M-step). In E-step, based on the current estimated $P(z \mid D)$ and $P(t \mid z)$, the posterior probability of $P(z \mid D,t)$ is computed for each document-word pair at each iteration. In M-Step, $P(z \mid D)$ and $P(t \mid z)$ are updated by maximizing (4), which will be used in the E-step of next iteration until convergence. Detailed steps of EM algorithm are discussed below:

Initialization: Define the maximum number of iterations *R*, and the number of latent topics *G* to be generated. For each document-topic and topic-word pair, assign random values to $P_0(z \mid D)$ and $P_0(t \mid z)$ between 0 and 1, with the constraints $\sum_z P_0(z \mid D) = 1$, and $\sum_t P_0(t \mid z) = 1$.

E-step: At iteration *r*, for each observed topic, word and document, z_p, t_b, and D_a, compute:

$$P_r(z_p \mid D_a, t_b) = \frac{P_{r-1}(t_b \mid z_p) P_{r-1}(z_p \mid D_a)}{\sum_z P_{r-1}(t_b \mid z) P_{r-1}(z \mid D_a)} \qquad (6)$$

where $P_{r-1}(z_p \mid D_a)$ and $P_{r-1}(t_b \mid z_p)$ are derived from iteration *r-1*.

M-step: At iteration *r*, for each document-topic and topic-word pair, compute $P_r(z_p \mid D_a)$ and $P_r(t_b \mid z_p)$ based on the following updating formulas:

$$P_r(t_b \mid z_p) = \frac{\sum_D n(D, t_b) P_r(z_p \mid D, t_b)}{\sum_{D,t} n(D, t) P_r(z_p \mid D, t)} \qquad (7)$$

$$P_r(z_p \mid D_a) = \frac{\sum_t n(D_a, t) P_r(z_p \mid D_a, t)}{\sum_t n(D_a, t)} \qquad (8)$$

More detailed derivation of (7) and (8) can be referred to [106,107,110].

The above E-step and M-step repeat until the maximum iteration *R*, or the log-likelihood function $L_r(\theta)$ in (1) met the criterion that $L_r(\theta) - L_{r-1}(\theta) \le \varepsilon$, where ε is set as the convergence goal of the model.

The output from this stage after EM learning that is used for building VSM model is a latent topic-document (LTD) matrix M_{td}, in which each document has a vector representation H_{D_l} that is mapped from indexed term space to latent topic space, $H_{D_l} = [P_{1,D_l}, P_{2,D_l}, ..., P_{G,D_l}]$, and $P_{i,D_l} = P_R(z_i \mid D_l)$, $i = 1,2,...G$, where R denotes the maximum number of iterations EM went through, and G denotes the number of topics generated. As a result, we have:

$$M_{td} = [H_{D_1}^T, H_{D_2}^T, ..., H_{D_N}^T].$$

4.1.2.2 Generate topic-document vector for previously unseen document

Although PLSA is originally designed for unsupervised learning, it can be extended to previously unseen (testing) documents. For a testing document D_u, we run through the same EM algorithm to generate the conditional probability of each latent topic z given D_u. However, during parameter estimation, all other parameters are kept fixed except $P(z \mid D_u)$. In initialization step, only $P_0(z \mid D_u)$ is assigned with random values between 0 and 1, with the constraints $\sum_z P_0(z \mid D_u) = 1$. In E-step, based on $P_{r-1}(z \mid D_u)$, the posterior probability of $P(z \mid D_u, t)$ is computed. In M-Step, only $P_r(z \mid D)$ is calculated by equation (8). Therefore, a vector representation H_{D_u} is generated for D_u, with the same dimension as H_{D_l}.

4.1.3 A VSM augmented with semi-supervised PLSA topic modeling

The PLSA algorithm introduced in the above section 4.1.2 generates latent topics by exploring the co-occurrence relationship of words in the document collection under a probabilistic framework, in order to discover the underlying semantic structure. However, it is originally designed for unsupervised learning, since it assume that none prior knowledge of the text is available. Therefore, documents in the same category might have different latent topic distribution due to different word occurrence. In the application of text categorization, usually we will have some information about the inter-connectivity between documents, such as category label, citation links and references, web page links and so forth [108,111]. As a result, a mixed probability model that couples the conditional probabilities for both words and inter-connectivity between documents could be extremely useful, in terms of providing more meaningful features and better understanding from text. The most well-known model that incorporates such information is proposed by Hoffman [108], which presents a joint probabilistic model of document content and connectivity. However, there are several issues needs to be solved. First of all, in the task of text categorization, usually only training documents have inter-connectivity available, that model is not able to model previously unseen documents. Secondly, the inter-connectivity variable, denoted as $P(c \mid z)$, increases the dimension of parameters need to be estimated, so that the efficiency of the system is decreased. Last but not least, the connection between documents should also be weighted, instead of simply using binary values (1 as connected, and 0 as not connected).

In this dissertation, we propose a semi-supervised PLSA algorithm that addresses the above issues, while incorporating the relationship between documents derived from both category labels and ontology networks. Details will be discussed both in this section and section 4.2.2.

4.1.3.1 Learning semi-supervised PLSA model from training documents

For a given latent topic, the probability of document connectivity is interpreted as the document's authority on that topic [108]. By introducing a joint probability model for document content and connectivity, as well as a hyper-weight α that balance the affection, the semi-supervised PLSA model is presented in Figure 19. Similarly as PLSA algorithm, the generative process of each observed document-word pair and connected document-document pair in the text corpus, *Tr*, is shown as following:

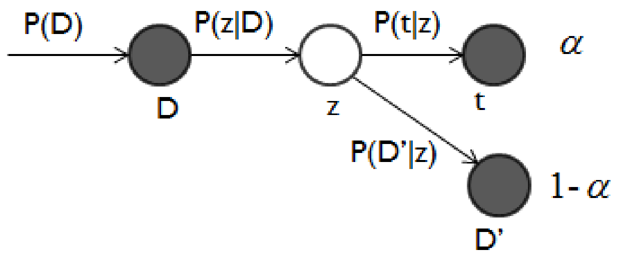


Figure 19. Graphical model representation of semi-supervised PLSA

1. Select a document *D* from *Tr* based on $P(D)$.
2. Pick a topic *z* according to $P(z \mid D)$.
3. Given *z*, generate a word t based on $P(t \mid z)$.
4. Given *z*, generate a document *D'*, based on $P(D' \mid z)$. This represents the probability of observing *D'* that is connected with *D*, given latent topic *z*.

The variables $P(z \mid D), P(t \mid z)$ and $P(D' \mid z)$ are what we want to estimate. As a result, we come up with the following joint log-likelihood function:

$$L_t = \sum_{D,t} n(D,t) \log(P(D) \sum_z P(t \mid z) P(z \mid D))$$
$$L_c = \sum_{D,D'} l(D,D') \log(P(D) \sum_z P(D' \mid z) P(z \mid D)), \qquad (9)$$
$$L = \alpha L_t + (1-\alpha) L_c,$$

Where $l(D,D')$ indicates whether document *D* and *D'* are connected ($l(D,D')=1$) or not ($l(D,D')=0$). For now, to simplify our problem, $l(D,D')$ is a binary value, and whether *D* is connected to *D'* or not is based on whether they both fall into the same category. $l(D,D')$ will be further updated by the word connection between documents in section 4.2.2.

Based on (9), we want to find $P(z|D)$, $P(t|z)$ and $P(D'|z)$ that maximize the log-likelihood function *L*. we then have:

$$\begin{aligned}&\arg\max(L) \propto \arg\max(\alpha L_t + (1-\alpha)L_c) = \arg\max(\alpha \sum_{D,t} n(D,t)\log(\sum_z P(t|z)P(z|D))\\ &+(1-\alpha)\sum_{D,D'} l(D,D')\log(\sum_z P(D'|z)P(z|D)))\end{aligned} \tag{10}$$

Similarly, we introduce $A_t(z)$ and $A_c(z)$ as two probability distributions of *z*, so that

$$\begin{aligned}&\arg\max(L) \propto \arg\max(\alpha \sum_{D,t} n(D,t)\log(\sum_z A_t(z)\frac{P(t|z)P(z|D)}{A_t(z)})\\ &+(1-\alpha)\sum_{D,D'} l(D,D')\log(\sum_z A_c(z)\frac{P(D'|z)P(z|D)}{A_c(z)})\end{aligned} \tag{11}$$

Based on the law of unconscious statistician, the above equation (11) yields to:

$$\begin{aligned}&(11) = \arg\max(\alpha \sum_{D,t} n(D,t)\log(E(\frac{P(t|z)P(z|D)}{A_t(z)}))\\ &+(1-\alpha)\sum_{D,D'} l(D,D')\log(E(\frac{P(D'|z)P(z|D)}{A_c(z)}))\end{aligned} \tag{12}$$

Using Jensen's Inequality, we find a lower bound function for the above equation (12), and when $\frac{P(t|z)P(z|D)}{A_t(z)} = C$ and $\frac{P(D'|z)P(z|D)}{A_c(z)} = C$, we have:

$$
\begin{aligned}
(12) &= \arg\max(\alpha \sum_{D,t} n(D,t) E(\log(\frac{P(t \mid z)P(z \mid D)}{A_t(z)})) \\
&+ (1-\alpha) \sum_{D,D'} l(D,D') E(\log(\frac{P(D' \mid z)P(z \mid D)}{A_c(z)})) \\
&= \arg\max(\alpha \sum_{D,t} n(D,t) \sum_z A_t(z)(\log(\frac{P(t \mid z)P(z \mid D)}{A_t(z)})) \\
&+ (1-\alpha) \sum_{D,D'} l(D,D') \sum_z A_c(z)(\log(\frac{P(D' \mid z)P(z \mid D)}{A_c(z)}))
\end{aligned} \qquad (13)
$$

Since we know that $\sum_z A_t(z) = 1$, $\sum_z A_c(z) = 1$, we have:

$$
A_t(z) = \frac{P(t \mid z)P(z \mid D)}{\sum_z P(t \mid z)P(z \mid D)} = P(z \mid t, D), \qquad (14)
$$

$$
A_c(z) = \frac{P(D' \mid z)P(z \mid D)}{\sum_z P(D' \mid z)P(z \mid D)} = P(z \mid D, D').
$$

Hence, $A_t(z)$ and $A_c(z)$ are posterior probabilities of z when we maximize (13), given the observation of each document-word pair and connected document-document pair, respectively. However, from the above equation (14) and (15), we could see the probability of $P(\mathrm{z} \mid D)$ and $P(D' \mid z)$ needs to be estimated separately for each document-topic pair, similar as the approach in [108]. It will be much easier if we could make some transformation so that we could estimate the same parameter instead. Based on the Bayes' Theorem, we have:

$$
\begin{aligned}
A_t(z) &= P(z \mid D, t) = \frac{P(t \mid z)P(D \mid z)P(z)}{\sum_z P(t \mid z)P(D \mid z)P(z)} \\
A_c(z) &= P(z \mid D, D') = \frac{P(D' \mid z)P(D \mid z)P(z)}{\sum_z P(D' \mid z)P(D \mid z)P(z)}
\end{aligned}. \qquad (15)
$$

This problem is then transferred into estimating $P(D \mid z)$, $P(t \mid z)$ and $P(z)$, which significantly reduces the size of parameters. Suppose the size of *Tr* is *N* and the number of topics generated is *G*, we now only need to estimate *G* additional parameters from $P(z)$ instead of $N \times G$ parameters from $P(z \mid D)$.

Similar to the PLSA, for semi-supervised PLSA, the maximization likelihood estimation in (9) can thus be solved using EM algorithm. During EM algorithm learning, in E-step, based on the current estimated $P(D \mid z)$, $P(t \mid z)$ and $P(z)$, the posterior probability of $P(z \mid D, t)$ and $P(z \mid D, D')$ is computed for each document-word pair and connected document-document pair at each iteration. In M-Step, $P(D \mid z)$, $P(t \mid z)$ and $P(z)$ are updated by maximizing (13), which will be used in the E-step of next iteration until convergence. Considering that $\sum_z P(z \mid D) = \frac{1}{P(D)} \sum_z P(D \mid z) P(z) = 1$, $P(D) = \sum_z P(D \mid z) P(z)$, the final conditional probability $P(z \mid D)$ for each document-topic pair can be calculated using: $P(z \mid D) = \frac{P(D \mid z) P(z)}{P(D)} = \frac{P(D \mid z) P(z)}{\sum_z P(D \mid z) P(z)}$, which means that $P(D)$ could be considered as a normalization constant for $P(z \mid D)$.

Computational steps of EM algorithm for semi-supervised PLSA

Initialization: Define the maximum number of iterations *R*, and number of topics *G* to be generated. For each document-topic and topic-word pair, assign random values to $P_0(D \mid z)$ and

$P_0(t \mid z)$ between 0 and 1, with the constraints $\sum_D P_0(D \mid z) = 1$, and $\sum_t P_0(t \mid z) = 1$. For each topic, initialize $P_0(z) = \frac{1}{G}$, which evenly distribute the probability of each topic at the beginning.

E-step: At iteration *r*, for each observed topic, word and document, z_p, t_b, D_a, and D_m that is connected to D_a, compute:

$$A_{t,r}(z_p) = P_r(z_p \mid D_a, t_b) = \frac{P_{r-1}(t_b \mid z_p)P_{r-1}(D_a \mid z_p)P_{r-1}(z_p)}{\sum_z P_{r-1}(t_b \mid z)P_{r-1}(D_a \mid z)P_{r-1}(z)} \qquad (16)$$

$$A_{c,r}(z_p) = P_r(z_p \mid D_a, D_m) = \frac{P_{r-1}(D_a \mid z_p)P_{r-1}(D_m \mid z_p)P_{r-1}(z_p)}{\sum_z P_{r-1}(D_a \mid z)P_{r-1}(D_m \mid z)P_{r-1}(z)} \qquad (17)$$

where $P_{r-1}(D_a \mid z_p)$, $P_{r-1}(D_m \mid z_p)$, $P_{r-1}(t_b \mid z_p)$ and $P_{r-1}(z_p)$ are derived from iteration *r-1*.

M-step: At iteration *r*, for each document-topic and topic-word pair, we want to calculate $P_r(D_a \mid z_p)$, $P_r(t_b \mid z_p)$ and $P_r(z_p)$ in order to maximize (13). As a result, at iteration *r*,

$$\begin{aligned}
(13) &= \arg\max(\alpha \sum_{D,t} n(D,t) \sum_z P_r(z \mid D,t)(\log(\frac{P_r(t \mid z)P_r(D \mid z)P_r(z)}{P_r(z \mid D,t)P_r(D)})) \\
&+ (1-\alpha)\sum_{D,D'} l(D,D') \sum_z P_r(z \mid D,D')(\log(\frac{P_r(D' \mid z)P_r(D \mid z)P_r(z)}{P(z \mid D,D')P_r(D)})) \qquad (18) \\
&\propto \arg\max \left(\begin{array}{l} \alpha \sum_{D,t} n(D,t) \sum_z P_r(z \mid D,t)\left[\log(P_r(t \mid z)) + \log(P_r(D \mid z)) + \log(P_r(z))\right] \\ + (1-\alpha)\sum_{D,D'} l(D,D') \sum_z P_r(z \mid D,D')\left[\log(P_r(D' \mid z)) + \log(P_r(D \mid z)) + \log(P_r(z))\right] \end{array} \right)
\end{aligned}$$

Let $(18) = \vartheta$, since we know that $\sum_t P_r(t \mid z) = 1$, $\sum_D P_r(D \mid z) = 1$ and $\sum_z P_r(z) = 1$, the above optimization problem could be solved using Lagrange multipliers [107], such that:

$$\vartheta - \sum_z \beta_z (\sum_t P_r(t \mid z) - 1) - \sum_z \omega_z (\sum_D P_r(D \mid z) - 1) - \delta(\sum_z P_r(z) - 1) = 0 \qquad (19)$$

Take derivative for (19) with respect to $P_r(t_b \mid z_p)$, $P_r(D_a \mid z_p)$ and $P_r(z_p)$ leads to the following stationary equations:

$$\beta_{z(p)} P_r(t_b \mid z_p) = \alpha \sum_D n(D, t_b) P_r(z_p \mid D, t_b) \qquad (20)$$

$$\omega_{z(p)} P_r(D_a \mid z_p) = \alpha \sum_t n(D_a, t) P_r(z_p \mid D_a, t) + 2 \times (1 - \alpha) \sum_{D'} l(D_a, D') P_r(z_p \mid D_a, D') \quad (21)$$

$$\delta P_r(z_p) = (\alpha - 1) \sum_{D,D'} l(D, D') P_r(z_p \mid D, D') \qquad (22)$$

Note here, $P_r(D_a \mid z)$ and $P_r(D' \mid z)$ are actually representing the same parameter, considering they are connected with each other and exchangeable. Therefore, in (21) they are merged together, so that $(1-\alpha)$ is multiplied by 2.

By summing up (20) by *t*, summing up (21) by *D*, and summing up (22) by *z*, we are able to solve the Lagrange multipliers $\sum_z \beta_z$, $\sum_D \omega_D$ and δ, and finally we get the updating equations for $P_r(t_B \mid z_p)$, $P_r(D_a \mid z_p)$ and $P_r(z_p)$:

$$P_r(t_b \mid z_p) = \frac{\sum_D n(D, t_b) P_r(z_p \mid D, t_b)}{\sum_{D,t} n(D, t) P_r(z_p \mid D, t)} \qquad (23)$$

$$P_r(D_a \mid z_p) = \frac{\alpha \sum_t n(D_a, t) P_r(z_p \mid D_a, t) + 2 \times (1 - \alpha) \sum_{D'} l(D_a, D') P_r(z_p \mid D_a, D')}{\alpha \sum_{D,t} n(D, t) P_r(z_p \mid D, t) + 2 \times (1 - \alpha) \sum_{D,D'} l(D_a, D') P_r(z_p \mid D, D')} \quad (24)$$

$$P_r(z_p) = \frac{\alpha \sum_{D,t} n(D,t) P_r(z_p \mid D,t) + 2 \times (1-\alpha) \sum_{D,D'} l(D,D') P_r(z_p \mid D,D')}{\alpha \sum_{D,t} n(D,t) + 2 \times (1-\alpha) \sum_{D,D'} l(D,D')} \quad (25)$$

The above E-step and M-step keep iterating until the maximum iteration *R*, or the log-likelihood function L_r in (9) met the criterion that $L_r - L_{r-1} \le \varepsilon$, where ε is set as the convergence goal of the model. The final conditional probability $P(z \mid D_l)$ for each topic given $D_l \in Tr$ can be calculated using: $P(z \mid D_l) = \frac{P(D_l \mid z)P(z)}{P(D_l)} = \frac{P(D_l \mid z)P(z)}{\sum_z P(D \mid z)P(z)}$.

Same as PLSA algorithm, the output from semi-supervised PLSA after EM learning is a semi-supervised topic-document (SSTD) matrix M_{sstd}, in which each document has a vector representation H_{D_l} that is mapped from indexed term space to latent topic space, $H_{D_l} = [P_{1,D_l}, P_{2,D_l}, ..., P_{G,D_l}]$, and $P_{i,D_l} = P_R(z_i \mid D_l)$, $i = 1,2,...G$, where *R* denotes the maximum number of iterations EM went through, and *G* denotes the number of topics generated. As a result, we have:

$$M_{sstd} = [H_{D_1}^T, H_{D_2}^T, ..., H_{D_N}^T].$$

4.1.3.2 Generate semi-supervised topic-document vector for previously unseen document

In order to extend semi-supervised PLSA to previously unseen (testing) documents, similar to PLSA, for a testing document D_u, we run through the EM algorithm to generate the conditional probability for D_u given latent topic *z*. During parameter estimation, all other parameters are kept fixed, except $P(D_u \mid z)$. However, for a unseen document D_u, $P(D_u \mid z)$ cannot be initialized directly. The following steps explain how EM algorithm is applied to D_u:

- In the initialization step, since $P(D_u \mid z) = \frac{P(z \mid D_u)P(D_u)}{P(z)} \propto \frac{P(z \mid D_u)}{P(z)}$, we assign random value to $P_0(z \mid D_u)$ between 0 and 1, with the constraint $\sum_z P_0(z \mid D_u) = 1$. Then $P_0(D_u \mid z)$ is calculated using $P(z)$ that has already been estimated during training, based on the following formula: $P_0(D_u \mid z) = \dfrac{P_0(z \mid D_u)\frac{1}{P(z)}}{\sum_z P_0(z \mid D_u)\frac{1}{P(z)}}$.
- In E-step, at iteration *r*, based on $P_{r-1}(D_u \mid z)$, the posterior probabilities, $P_r(z \mid D_u, t)$ and $P_r(z \mid D_u, D')$ are computed, where $D' \in Tr$, and D' is connected to D_u.
- In M-Step, only $P_r(D_u \mid z)$ is calculated by equation (24).
- The final conditional probability $P(z \mid D_u)$ for each topic given D_u can be calculated using: $P(z \mid D_u) = \dfrac{P(D_u \mid z)P(z)}{P(D_u)} = \dfrac{P(D_u \mid z)P(z)}{\sum_z P(D_u \mid z)P(z)}$.

Therefore, a vector representation H_{D_u} is generated for D_u, with the same dimension as H_{D_l}.

It is obvious that there is a problem about how to find out what D' in *Tr* is connected to the testing document, and how. This is done by using word semantic information extracted from ontology network, which will be discussed in detail in section 4.2.2. As a result, the system framework of using semi-supervised PLSA is shown in Fig. 20, where the VSM augmented with

PLSA topic modeling also takes the CD matrix generated from VSM augmented with WordNet ontology.

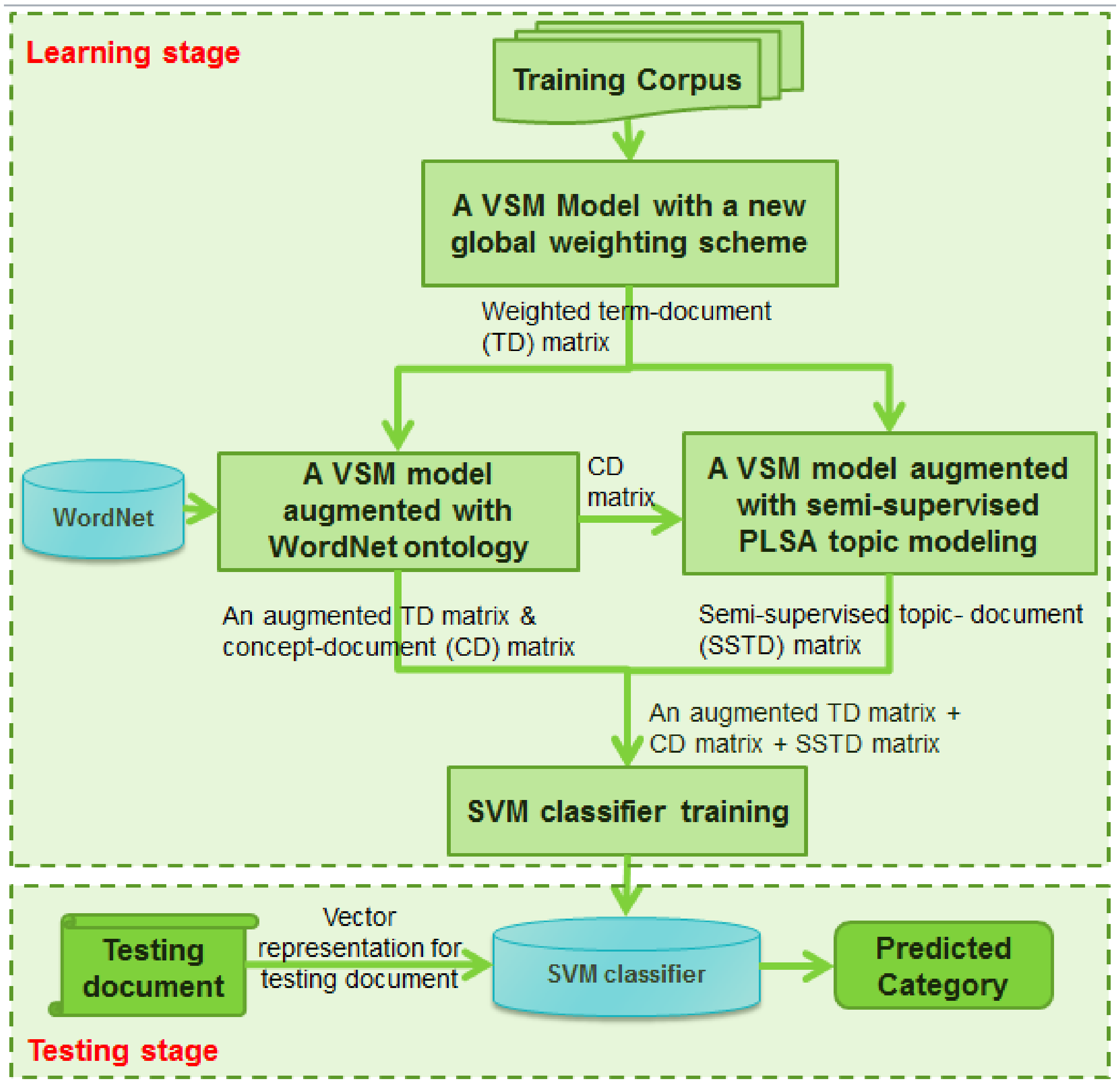

Learning stage
Training Corpus
A VSM Model with a new global weighting scheme
Weighted term-document (TD) matrix
WordNet
A VSM model augmented with WordNet ontology
CD matrix
A VSM model augmented with semi-supervised PLSA topic modeling
An augmented TD matrix & concept-document (CD) matrix
Semi-supervised topic- document (SSTD) matrix
An augmented TD matrix + CD matrix + SSTD matrix
SVM classifier training
Testing document
Vector representation for testing document
SVM classifier
Predicted Category
Testing stage

Figure 20. Proposed text categorization model framework

The above discussion in section 4.1 mainly focuses on how to generate latent semantic features from text documents, which are incorporated into conventional VSM model. The connectivity between documents, together with the relationship between words, allows us to generate a mixed joint probabilistic model for a text collection that provides a solid foundation for an accurate and meaningful text representation.

4.2 A VSM augmented with WordNet ontology

Word ontology networks, as discussed in section 2.3, provide semantic word relationships that could be utilized to facilitate text mining applications. This section presents the detail of building text categorization model using WordNet ontology network, in terms of generating an augmented TD matrix and a "concept" level CD matrix.

4.2.1 An augmented TD matrix generated using WordNet

In our proposed text categorization model, WordNet is used in two ways, derived from and modified based on basic approaches introduced in [46]: "Add" and "Replace" rules, considering the "Concept only" rule lost single term information, and did not achieve as good performance as other two rules in text clustering tasks, as reported in [46]. This answers the question in section 2.5.3 about which rules should be selected to apply to VSM features.

For each indexed term t_i generated by VSM model, we first use POS tagging such as Stanford POS tagger [112] to identify its lexical category. After that, t_i is feed into WordNet ontology to find its list of synonyms, S_j. Note here, word sense disambiguation (WSD) can be applied to obtain more accurate synonym generation, however it is beyond our research scope, and it is not our intention to find a most appropriate WSD model. As a result, we find the synset S_j based on the first meaning of t_i, to simplify our problem.

In some applications, POS tagging may not be very reliable, e.g., text documents are noisy and lack of grammar structure and sentence boundary [5]. In those cases, we generate the synset for t_i only considering one word class from "Noun", "Verb" or "Adjective", and choose the best one based on system performance evaluation. Through our experiment we found out that "Noun" synsets always have the best accuracy, which will be presented in Chapter 5.

The above discussion answers the question raised in section 2.5.3 about determining the scope of synset for a word with multiple word categories.

Although in WordNet, the major relationship is synonymy, we also find out that most synsets are connected to other synsets via a number of semantic relations. These relations vary based on the type of word. For "Noun" synsets, the relations mainly include hypernym/hyponym (word A is a kind of word B or vice versa, e.g., dog vs. canine), and meronym/holonym (word A is a part of word B or vice versa, e.g., window vs. building), which are our major focus in this dissertation. For "Verb" or "Adjective" synsets, we only consider the synonymy relation. This solves the issue of making full use of word relationships other than synonymy, as mentioned in section 2.5.3.

The ultimate goal of finding the related synsets for a given term, including synonymy, hypernym/hyponym and meronym/holonym, is to find out all terms within these synsets, and use this term relationship information to augment VSM model. The following Fig. 21 illustrates an example of generating the list of related words, *L_syn*, in WordNet for an indexed term t_i, especially when t_i is a noun. Here, $t_{x,y}$ denotes the y^{th} term included in the x^{th} synset. Starting from S_j, denoted as root level 0, we find out its hypernym/hyponym and meronym/holonym synsets, extract all unique terms included in these synsets, and add them into *L_syn*. The next level starts from synsets S_a, S_b, S_c and S_d, and their hypernym/hyponym and meronym/holonym synsets are extracted respectively. This procedure continues to explore through the entire "graph" of S_j until there are no synsets related to the current synset. All unique terms included in the list of related synsets generated for S_j are then added into L_syn.

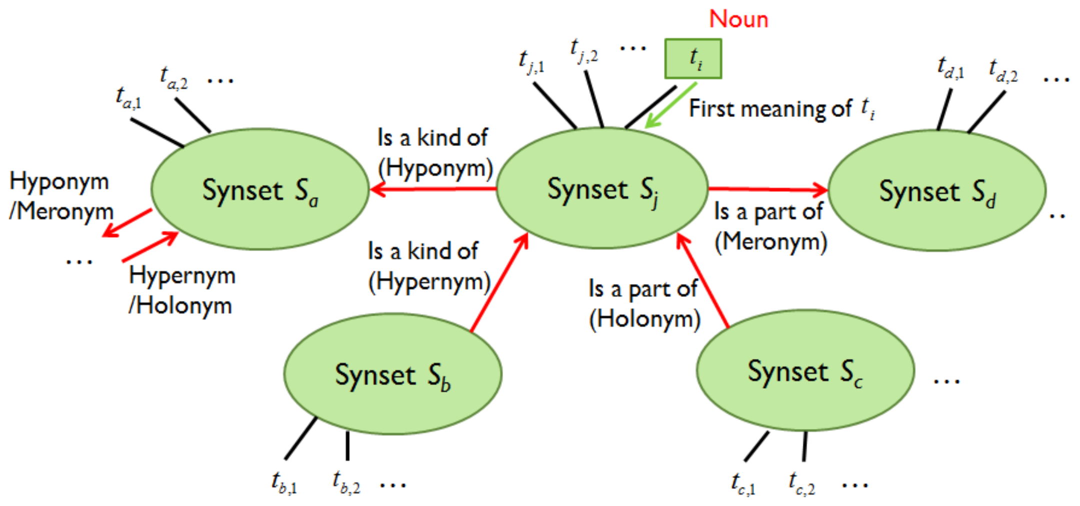


Figure 21. Example of generating related words in WordNet

In order to build a hierarchical semantic relationship between synsets, we assign a weight for each edge in the graph generated for S_j. The basic idea here is that, terms found in different level of synsets, should be assigned with different semantic weight, the deeper the synset level is, the lower weight we are expecting. One example is shown in Fig. 22. Two coefficients, $\mu \in (0,1]$ and $\nu \in (0,0.5]$, are defined to represent the weight of the edge for hypernym/hyponym and meronym/holonym relationships, respectively. Here, considering that meronym/holonym relationship is less significant than hypernym/hyponym in terms of semantic similarity, e.g., document talks about "window" might not have any relationship with document talks about "building", we use $\nu = \frac{1}{2}\mu$ throughout our experiments.

After assigning weight to each edge, we then assign a weight for each synset that is related to S_j, in order to reflect the weight decrease along the path from S_j to its related synsets. Starting from the root synset S_j, its weight, denoted as Ψ_{S_j}, equals to 1. After that, the weight Ψ_{S_x} for any synset S_x that is related to S_j, is calculated by multiplying the weights of all the edges along the shortest path from S_x to S_j. If multiple paths are found, then the maximum value is selected for Ψ_{S_x}. For example, in Fig. 22, $\Psi_{S_f} = \mu^2$ for synset S_f, $\Psi_{S_e} = \mu \times \nu$ for synset S_e.

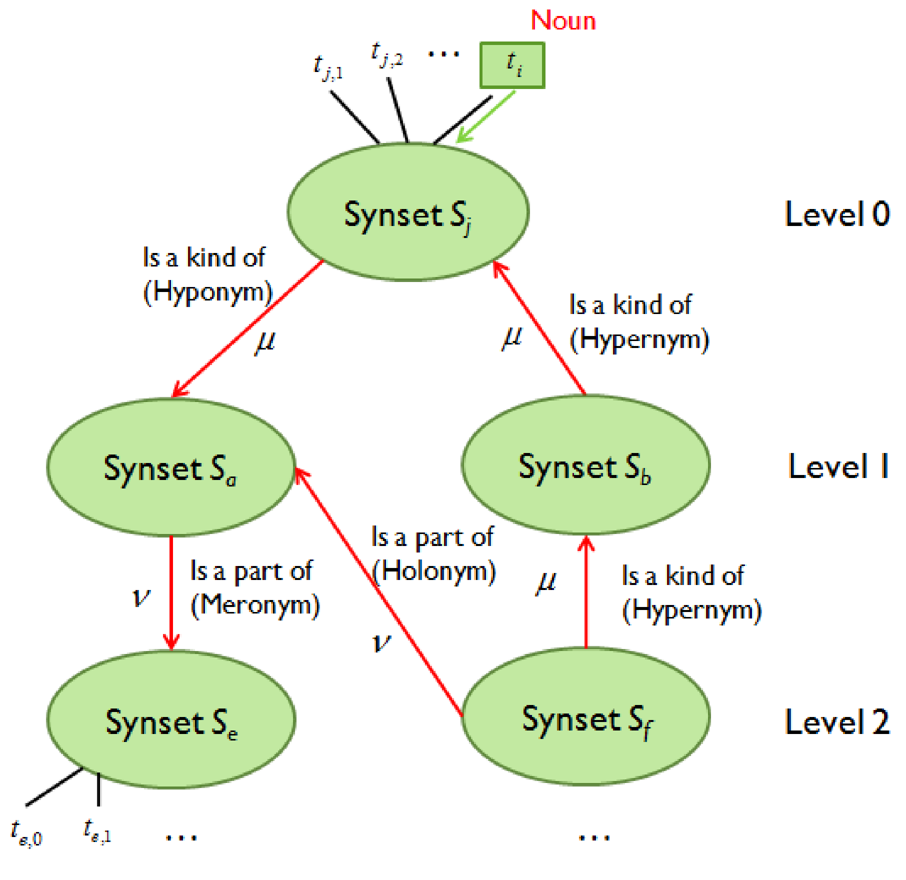


Figure 22. Example of weighting edges in the tree structure generated for synset

4.2.1.1 Generate concept-document (CD) matrix ("add" rule)

Under this rule, a *CD* matrix M_c is generated using WordNet by introducing the "concept" level features, which represents the *V* related groups of terms generated from the term list *T_L* of document collection *Tr*, denoted as $L\_syn_i, i = 1,2,...V$. Mathematically, for a document $D_l \in Tr$, its "concept" vector representation Q_{D_l} is defined as following:

$$Q_{D_l} = [q_{1,D_l}, q_{2,D_l}, ..., q_{V,D_l}],\ q_{i,D_l} = \sum_{t_r} (w_{t_r,D_l} \times \Psi_{S_x}), t_r \in L\_syn_i, t_r \in S_x, i = 1,2,...,V\,, \qquad (26)$$

where *V* represents the total number of synsets generated from the term list *T_L*, q_{i,D_l} denotes the weight of "concept" L_syn_i , which is calculated by first multiplying weighted term frequency value w_{t_r,D_l} (e.g., *tf-idf*) for each term t_r in L_syn_i with the weight of synset S_x that includes t_r , and then summing them together. For example, in the following Fig. 23, assume the concept L_syn_1 generated from term "car" contains the following terms: "car", "motorcar", "motorbus", "bus", "minibus", "window" and "quarterlight", therefore, for a document $D_l \in Tr$,

$$\begin{aligned} q_{1,D_l} &= w_{car,D_l} + w_{motorcar,D_l} + \mu \times (w_{bus,D_l} + w_{motorbus,D_l}) \\ &+ \nu \times w_{window,D_l} + \mu \times \nu \times w_{quarterlight,D_l} + \mu^2 \times w_{\mathrm{minibus},D_l} \end{aligned}.$$

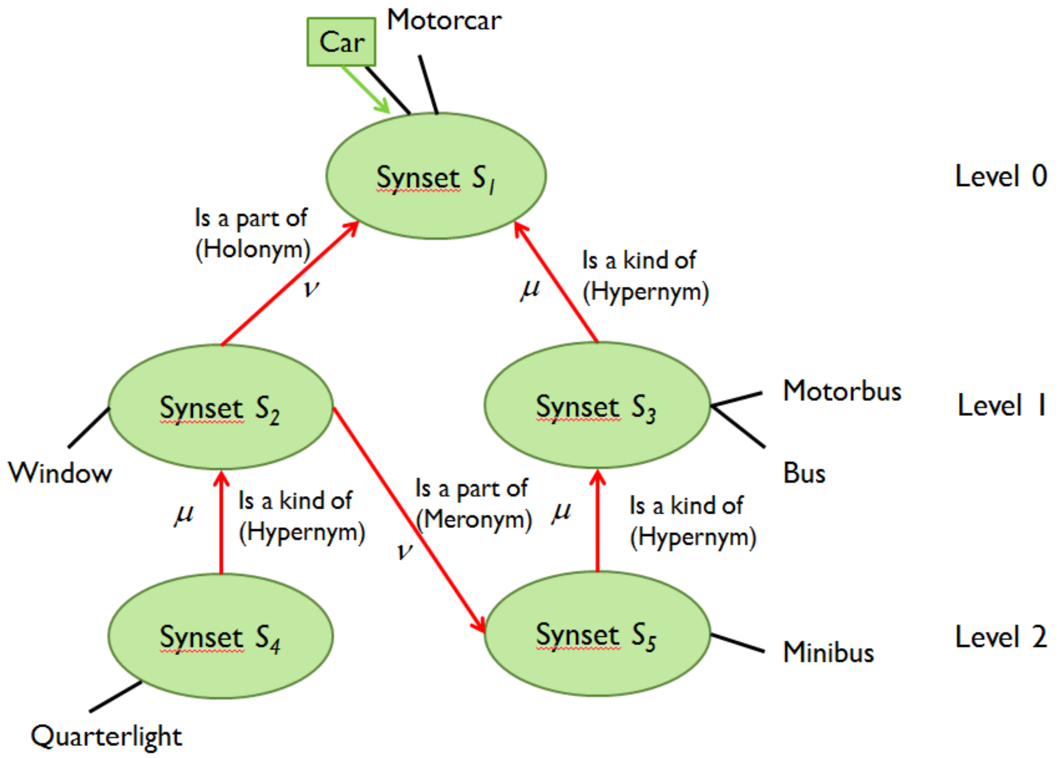


Figure 23. Example of concept generation in "add" rule

This procedure makes the full use of synonymy, hypernym/hyponym and meronym/holonym to assign an appropriate weight for each "concept" features used for augmented VSM model. It answers the question mentioned in section 2.5.3 about what value should be assigned to the concept features added to TD matrix. Thus, for the output of this stage, we have:

$$M_c = [Q_{D_1}^T, Q_{D_2}^T, \ldots, Q_{D_N}^T].$$

Similarly, for a previously unseen document D_u, we generate the "concept" vector representation Q_{D_u}, $Q_{D_u} = [q_{1,D_u}, q_{2,D_u}, \ldots, q_{V,D_u}]$, for the testing purpose.

4.2.1.2 Generate augmened TD matrix (modified "repl" rule)

Under this rule, the term-document matrix M_0 in section 4.1.1 is replaced by a new term-document matrix generated using WordNet, in a way that for a term t_i having synset S_i , its weight in document D_l is updated using the following equation:

$$w'_{t_i,D_l} = \max(\forall w_{t_r,D_l} \left| t_r \in S_i, t_r \in T\_L \right.) .$$

The above equation ensures that semantically similar terms share the same weighting value, so that they are considered as equally important. For example, if term t_x = "entire" appears in document *A* and t_y = "total" appears in document *B*, and suppose $S_x = S_y = \{t_x, t_y\}$, then we will have $w'_{x,A} = w'_{y,A} = w'_{x,B} = w'_{y,B}$. The output from this stage is a TD matrix M_1 that has the same dimension as M_0, where $M_1 = [W'^T_{D_1}, W'^T_{D_2}, ..., W'^T_{D_N}]$, and W'_{D_i} denotes the vector representation for document D_i in *Tr*, $W'_{D_l} = [w'_{t_1,D_l}, w'_{t_2,D_l}, ..., w'_{t_K,D_l}]$.

4.2.2 Generate document-document connection for semi-supervised PLSA using WordNet

We mentioned in section 4.1.1.2 that when applying semi-supervised PLSA to previously unseen document D_u , there is a problem about how to find out what documents in *Tr* are connected to the testing document, and how. This leads to the problem of determining $l(D_u, D)$ as shown in the log-likelihood function in (9), which indicates whether D_u and *D* are connected or not. From the perspective of joint probability of all observed documents pairs that are connected with each other, we propose the following approach of generating $l(D_u, D)$ for the pair of D_u and each $D \in Tr$.

Suppose we have *V* related "concepts" generated from the term list *T_L* of document collection *Tr*, denoted as $L\_syn_i, i = 1,2,...V$:

- Extract weighted frequency based vector representation for D_u and *D*, denoted as W_{D_u} and W_D, as discussed in section 4.1.1, where $W_D \in M_0$.
- Generate a sub-list if *T_L*, denoted as *T_L_sub* that has *O* terms, where for each term $t_o \in T\_L\_sub, o = 1,2,...O$, t_o does not belong to any of the *V* "concepts".
- Generate "concept" vector representation for D_u and *D*, denoted as Q_{D_u} and Q_D, respectively, based on equation (26), where $Q_D \in M_c$.
- The connection value $l(D_u, D)$ between D_u and *D* is thus calculated as following:

  $l(D_u, D) = \sum_{j=1}^{V} \min(Q_{D_u}(j), Q_D(j)) + \sum_{k=1}^{O} \min(W_{D_u}(k), W_D(k))$, where $Q_D(j)$ denotes the *j*th concept in vector Q_D, and $W_D(k)$ denotes the *k*th weighted term frequency in *T_L_sub*.

  The basic idea here is that, $l(D_u, D)$ is the "weighted frequency" that we observe both D_u and *D* have concept occurrence or "non-concept" term occurrence, which represents the connection value between D_u and *D*.

After generating $l(D_u, D)$ for the pair of D_u and *D*, it is obvious that $l(D, D')$ generated for training document collection *Tr* should also be updated. Instead of using binary values ($l(D, D') = 1$ or $l(D, D') = 0$), for all pair of *D* and *D'*, where both $D, D' \in Tr$, if $l(D, D') = 1$, $l(D, D')$ is updated using the same approach discussed above. With the procedures discussed above, we are able to apply semi-supervised PLSA model for both training set *Tr* and unseen document D_u, by generating the semi-supervised topic-document (SSTD) matrix M_{sstd}, and a vector representation H_{D_u}, respectively.

4.3 A step-by-step example of proposed text representation model generation procedure

To provide a more comprehensive illustration of how the above section 4.1 and 4.2 works, we hereby designed a “toy” dataset that is derived from [113] and walk through the whole procedure of VSM model generation using PLSA, semi-supervised PLSA and WordNet ontology. This dataset is named as Human Computer Interface and Graph Theory (HCI_GT).

The HCI_GT contains 9 documents as training data separated into two categories, and one “unseen” document D_u for testing, defined as following, where bolded words denote indexed terms:

Category A: Human Computer Interface (HCI)   Category B: Graph Theory(GT)

*A1: **Human** machine **Interface** for ABC **computer** applications*

*A2: A **survey** of **user** opinion of **computer system response time***

*A3: The **EPS user interface** management **system***

*A4: **System** and **human system** engineering testing of **EPS***

*A5: Relation of **user** perceived **response time** to error management*

*B1: **System** of random, binary, ordered **tree***

*B2: The intersection **graph** of paths in **tree***

*B3: **Graph minors** IV: Widths of **tree** and well-quasi-ordering*

*B4: **Graph minors**: A **study***

*D_u: A **survey** of decision **tree system***

4.3.1 *Build VSM model with CE_W global weighting scheme*

Hence, in this training set *Tr*, $N = 9$, $K = 13$. As discussed in section 4.1.1, the vector representation generated for each document in *Tr* and for testing document D_u, based on term frequency (*tf*) with *idf* global weighting and *tf* with *CE_W* global weighting, are shown in the following Table 14 - 15.

Table 14 *Tf-idf* representation for HCI_GF

	human	interface	computer	user	system	response	time	EPS	survey	trees	graph	minors	study
A1	0.577	0.577	0.577	0	0	0	0	0	0	0	0	0	0
A2	0	0	0.410	0.299	0.221	0.410	0.410	0	0.598	0	0	0	0
A3	0	0.595	0	0.435	0.321	0	0	0.595	0	0	0	0	0
A4	0.594	0	0	0	0.542	0	0	0.594	0	0	0	0	0
A5	0	0	0	0.459	0	0.628	0.628	0	0	0	0	0	0
B1	0	0	0	0	0.594	0	0	0	0	0.805	0	0	0
B2	0	0	0	0	0	0	0	0	0	0.707	0.707	0	0
B3	0	0	0	0	0	0	0	0	0	0.508	0.508	0.696	0
B4	0	0	0	0	0	0	0	0	0	0	0.381	0.522	0.763
D_u	0	0	0	0	0.313	0	0	0	0.849	0.425	0	0	0

Table 15 *Tf-CE_W* representation for HCI_GF

	human	interface	computer	user	system	response	time	EPS	survey	trees	graph	minors	study
A1	1	1	1	0	0	0	0	0	0	0	0	0	0
A2	0	0	1	1	0.126	1	1	0	1	0	0	0	0
A3	0	1	0	1	0.126	0	0	1	0	0	0	0	0
A4	1	0	0	0	0.213	0	0	1	0	0	0	0	0
A5	0	0	0	1	0	1	1	0	0	0	0	0	0
B1	0	0	0	0	0.126	0	0	0	0	1	0	0	0
B2	0	0	0	0	0	0	0	0	0	1	1	0	0
B3	0	0	0	0	0	0	0	0	0	1	1	1	0
B4	0	0	0	0	0	0	0	0	0	0	1	1	1
D_u	0	0	0	0	0.126	0	0	0	1	1	0	0	0

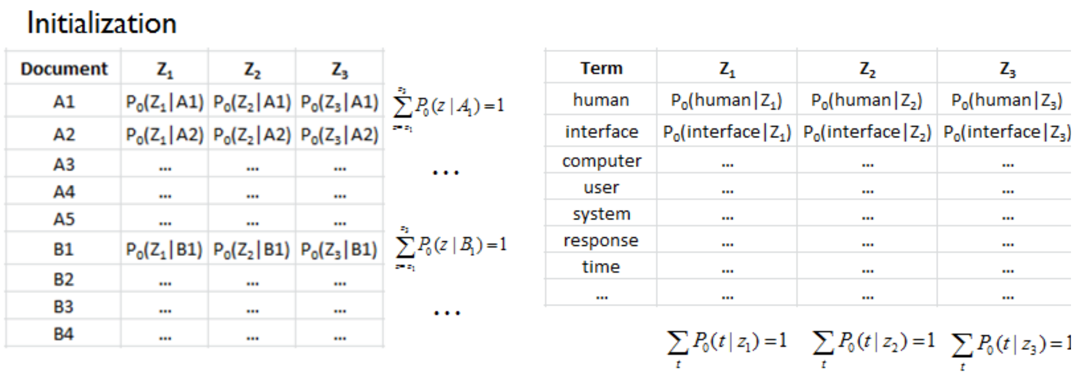
Comparing with *idf* global weighting, when applying *CE_W*, the word "system" is assigned with a lower weight because that it occurs in both category *A* and *B* and therefore not as important as others in terms of differentiating documents in *A* and *B* categories. All other terms are assigned with global weight equals to 1.

4.3.2 Build VSM model with PLSA topic modeling

Following the EM algorithm for PLSA discussed in section 4.1.1.1, we first define the number of maximum iteration R = 500, and ε = 1E-5. We found out that in our empirical case studies, generally speaking, the number of latent topics should be defined as around $\sqrt{K}$, where K is the number of indexed terms. We will further discuss the effect of selecting different number of topics in section 5.3. Here, because that the HCI_GT has 9 indexed terms, it is reasonable to define $G = 3$. The three latent topics are denoted as z_1, z_2 and z_3, respectively.

The steps of building LTD matrix for training documents in *Tr* are described as following:

- At the initialization step, for each document-topic and topic-word pair, assign random values to $P_0(z \mid D)$ and $P_0(t \mid z)$ between 0 and 1, with the constraints $\sum_{z=z_1}^{z_3} P_0(z \mid D) = 1$, and $\sum_t P_0(t \mid z) = 1, z = z_1, z_2, z_3$. As a result, we have the following two matrixes, M_{P_zD} and M_{P_tz}, as shown in the following Fig. 24:



Initialization

Document	Z_1	Z_2	Z_3
A1	$P_0(Z_1\|A1)$	$P_0(Z_2\|A1)$	$P_0(Z_3\|A1)$
A2	$P_0(Z_1\|A2)$	$P_0(Z_2\|A2)$	$P_0(Z_3\|A2)$
A3	...	...	...
A4	...	...	...
A5	...	...	...
B1	$P_0(Z_1\|B1)$	$P_0(Z_2\|B1)$	$P_0(Z_3\|B1)$
B2	...	...	...
B3	...	...	...
B4	...	...	...

$\sum_{z=z_1}^{z_3} P_0(z \mid A_1) = 1$

...

$\sum_{z=z_1}^{z_3} P_0(z \mid B_1) = 1$

...

Term	Z_1	Z_2	Z_3
human	$P_0(\text{human}\|Z_1)$	$P_0(\text{human}\|Z_2)$	$P_0(\text{human}\|Z_3)$
interface	$P_0(\text{interface}\|Z_1)$	$P_0(\text{interface}\|Z_2)$	$P_0(\text{interface}\|Z_3)$
computer	...	...	...
user	...	...	...
system	...	...	...
response	...	...	...
time	...	...	...
...	...	...	...

$\sum_t P_0(t \mid z_1) = 1$ $\sum_t P_0(t \mid z_2) = 1$ $\sum_t P_0(t \mid z_3) = 1$

Figure 24. Example of EM algorithm Initialization

- At E-step (expectation step), at iteration r, $P_r(z_1 \mid D,t), P_r(z_2 \mid D,t), P_r(z_3 \mid D,t)$ are calculated for each term-document pair using equation (6), which use the $P_{r-1}(z \mid D)$ and $P_{r-1}(t \mid z)$ initialized before. Fig. 25 shows this E-step process in iteration 1.

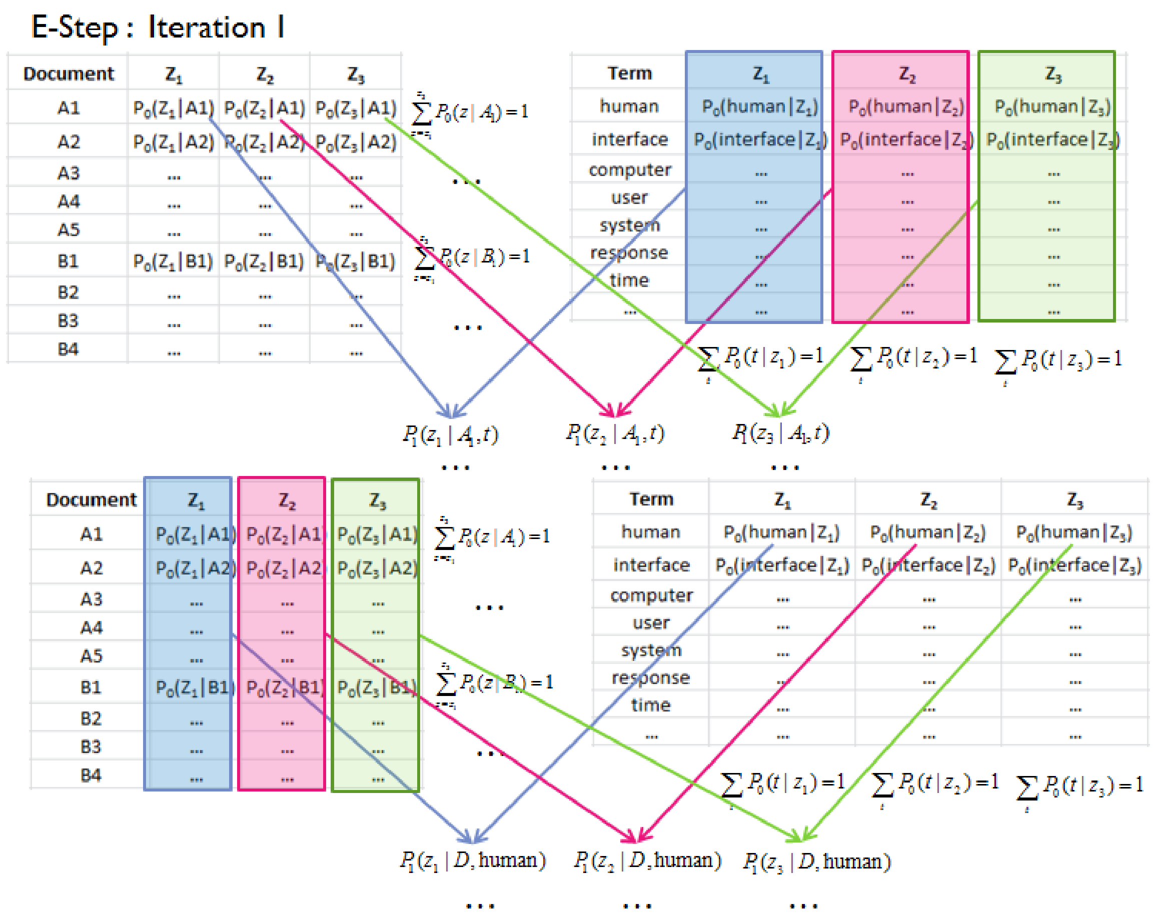
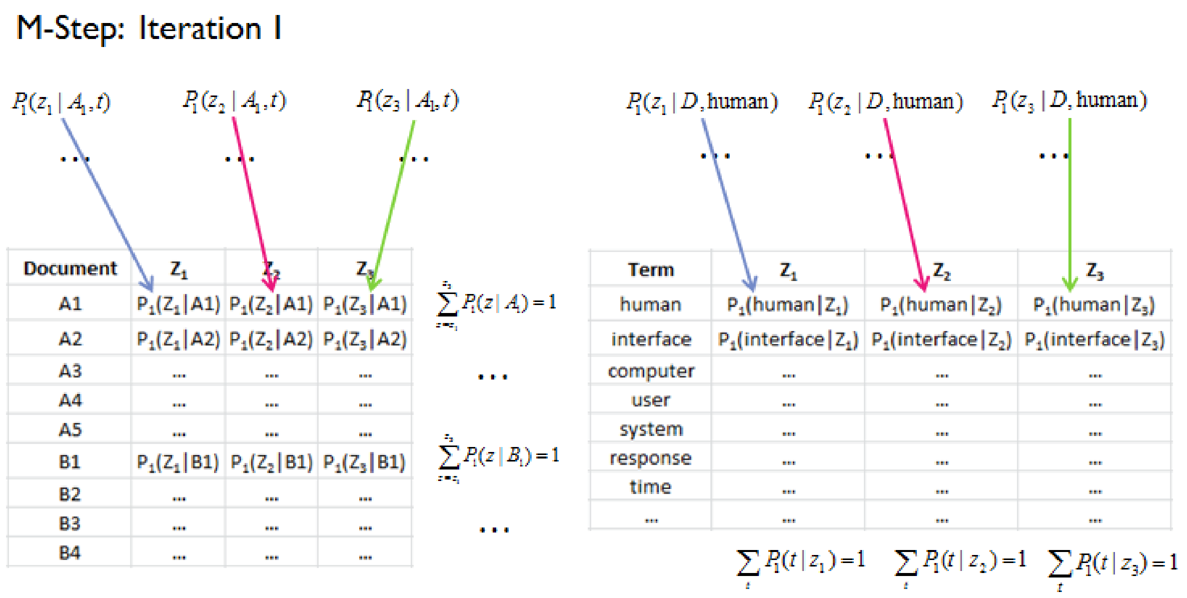


Figure 25. Example of EM algorithm E-Step

- At M-step (maximization step), at iteration *r*, for each document-topic and topic-word pair, compute $P_r(z|D)$ and $P_r(t|z)$ based on the updating formulas in (7) and (8). Apparently, this step uses the calculated conditional posterior probability $P_r(z|D,t)$ to update the probability $P_{r-1}(z|D)$ and $P_{r-1}(t|z)$ into $P_r(z|D)$ and $P_r(t|z)$, Fig. 26 shows this M-step process in iteration 1, in M_{P_zD} and M_{P_tz}.


M-Step: Iteration I

$P_1(z_1 | A_1, t)$ $P_1(z_2 | A_1, t)$ $P_1(z_3 | A_1, t)$

Document	Z₁	Z₂	Z₃
A1	P₁(Z₁|A1)	P₁(Z₂|A1)	P₁(Z₃|A1)
A2	P₁(Z₁|A2)	P₁(Z₂|A2)	P₁(Z₃|A2)
A3	...	...	...
A4	...	...	...
A5	...	...	...
B1	P₁(Z₁|B1)	P₁(Z₂|B1)	P₁(Z₃|B1)
B2	...	...	...
B3	...	...	...
B4	...	...	...

$\sum_{z=z_1}^{z_3} P_1(z | A_1) = 1$ ... $\sum_{z=z_1}^{z_3} P_1(z | B_1) = 1$ ...

$P_1(z_1 | D, \text{human})$ $P_1(z_2 | D, \text{human})$ $P_1(z_3 | D, \text{human})$

Term	Z₁	Z₂	Z₃
human	P₁(human|Z₁)	P₁(human|Z₂)	P₁(human|Z₃)
interface	P₁(interface|Z₁)	P₁(interface|Z₂)	P₁(interface|Z₃)
computer	...	...	...
user	...	...	...
system	...	...	...
response	...	...	...
time	...	...	...
...	...	...	...

$\sum_t P_1(t | z_1) = 1$ $\sum_t P_1(t | z_2) = 1$ $\sum_t P_1(t | z_3) = 1$

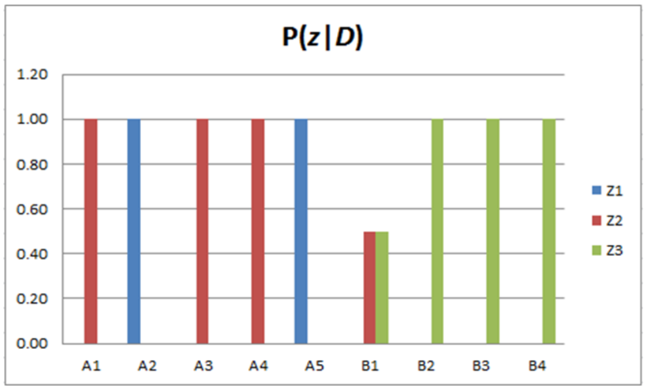
Figure 26. Example of EM algorithm M-Step

- The above E-step and M-step repeat until the maximum iteration R, or the log-likelihood function $L_r(\theta)$ in (1) met the criterion that $L_r(\theta) - L_{r-1}(\theta) \leq \varepsilon$, at iteration r. Here, we denote the final estimation for each topic-word pair after the above training process as $P_{Tr}(t | z)$.

The final results of M_{P_zD} and M_{P_tz} after convergence, with size 9×3 and 13×3 respectively, is shown in Fig. 27.

M_{P_zD}

Z_1	Z_2	Z_3	Document
0.00	1.00	0.00	A1
1.00	0.00	0.00	A2
0.00	1.00	0.00	A3
0.00	1.00	0.00	A4
1.00	0.00	0.00	A5
0.00	0.50	0.50	B1
0.00	0.00	1.00	B2
0.00	0.00	1.00	B3
0.00	0.00	1.00	B4



M_{P_tz}

Z_1	Z_2	Z_3	Term
0.000	0.167	0.000	human
0.000	0.167	0.000	interface
0.111	0.083	0.000	computer
0.222	0.083	0.000	user
0.111	0.333	0.000	system
0.222	0.000	0.000	response
0.222	0.000	0.000	time
0.000	0.167	0.000	EPS
0.111	0.000	0.000	survey
0.000	0.000	0.333	tree
0.000	0.000	0.333	graph
0.000	0.000	0.222	minors
0.000	0.000	0.111	study

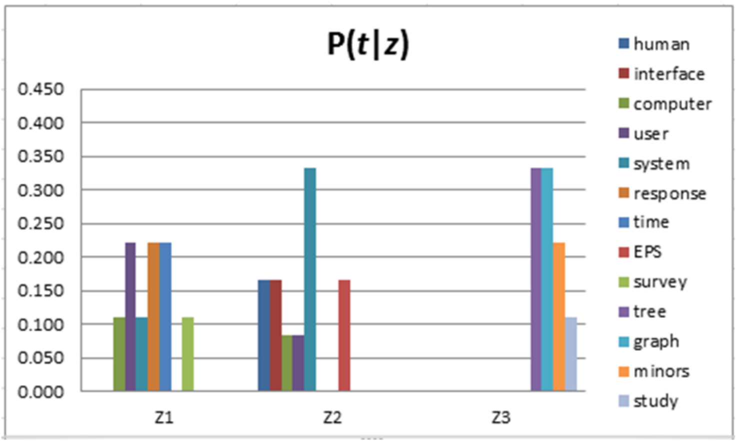


Figure 27. Example of EM algorithm result after convergence

In M_{P_zD}, latent topic z_1 and z_2 can be considered as "sub-categories" in category A, and most of category B documents only have occurrence of latent topic z_3, except document B_1, which contains both term "system" and "tree", therefore both z_2 and z_3 have 0.5 probability of occurrence. In M_{P_tz}, it is obvious that latent topic z_3 contains all terms that only occurs in category B, and term "computer", "user" and "system" all have probability on latent topic z_1 and z_2.

For testing document D_u, we run through the EM algorithm to generate the conditional probability of each latent topic z for D_u using the following procedure with the same parameter setting for R, G and ε:

- At the initialization step, for each topic z for the given D_u, assign random values to $P_0(z \mid D_u)$ between 0 and 1, with the constraints $\sum_{z=z_1}^{z_3} P_0(z \mid D_u) = 1$.
- At E-step, at iteration r, $P_r(z_1 \mid D_u, t), P_r(z_2 \mid D_u, t), P_r(z_3 \mid D_u, t)$ are calculated using equation (6) and $P_{Tr}(t \mid z)$ from the training process..
- At M-step, at iteration r, compute $P_r(z \mid D_u)$ based on the updating formula in (8).
- The above E-step and M-step repeat until the maximum iteration R, or the log-likelihood function $L_r(\theta)$ in (1) met the criterion that $L_r(\theta) - L_{r-1}(\theta) \le \varepsilon$, at iteration r. Here, we denote the final estimation for each topic-D_u pair as $P_F(z \mid D_u)$.

Therefore, for testing document D_u, the final topic-document representation is an 1×3 vector V_{D_u}, where $V_{D_u} = [P_F(z_1 \mid D_u), P_F(z_2 \mid D_u), P_F(z_3 \mid D_u)]$, as shown in Fig. 28.

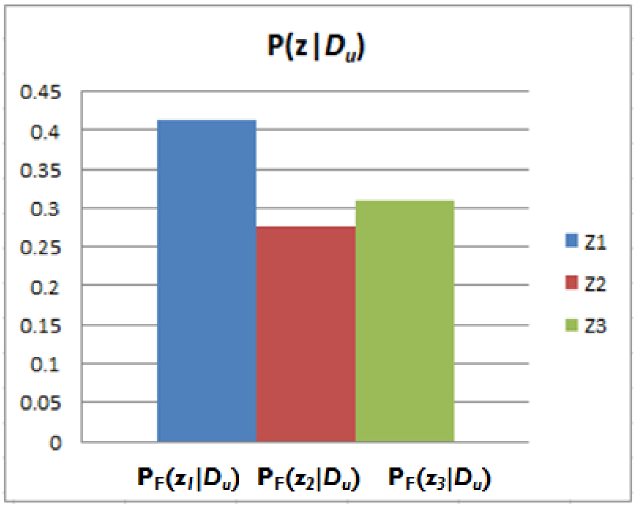


Figure 28. EM algorithm result for testing document D_u

4.3.3 Build VSM model with WordNet ontology

In this example dataset, HCI_GT, two terms could be found in WordNet as synonym: "survey" and "study". These two terms occurs in document A_2, B_4 and testing document D_u. No other hypernym/hyponym and meronym/holonym relationship is found in HCI_GT.

As discussed in section 4.2.1.1, based on "add" rule, the *CD* matrix M_c generated for training documents in *Tr* and the "concept" vector representation Q_{D_u} generated for testing document D_u, using *CE_W* global weighting, therefore only have one "concept" that contains "survey" and "study", as shown in Table 16, where $M_c = \left[Q_{A_1}^T, Q_{A_2}^T, \ldots, Q_{B_4}^T\right] = \begin{bmatrix} q_{1,A_1} \\ q_{1,A_2} \\ \ldots \\ q_{1,B_4} \end{bmatrix}$, $q_{1,D_l} = w_{\text{survey},D_l} + w_{\text{study},D_l}$, and $Q_{D_u} = q_{1,D_u} = w_{\text{survey},D_u} + w_{\text{study},D_u}$.

Table 16 CD matrix generation for *Tr* and D_u

Document	Concept (survey, study)
A_1	0
A_2	1
A_3	0
A_4	0
A_5	0
B_1	0
B_2	0
B_3	0
B_4	1
D_u	1

As discussed in section 4.2.1.2, based on "replace" rule, the vector representation generated for HCI_GT based on *CE_W* global weighting is updated using the following equation:

$w'_{\text{survey},D} = w'_{study,D} = \max\left(w_{\text{survey},D}, w_{\text{study},D}\right)$. The augmented TD vector for each document in HCI_GT is shown in the following Table 17, where both "survey" and "study" has weight equals to 1 in document A_2, B_4 and D_u:

Table 17 *Tf-CE_W* representation for *Tr* and D_u after using WordNet "replace" rule

	human	**interface**	**computer**	**user**	**system**	**response**	**time**	**EPS**	**survey**	**trees**	**graph**	**minors**	**study**
A_1	1	1	1	0	0	0	0	0	0	0	0	0	0
A_2	0	0	1	1	0.126	1	1	0	1	0	0	0	1
A_3	0	1	0	1	0.126	0	0	1	0	0	0	0	0
A_4	1	0	0	0	0.213	0	0	1	0	0	0	0	0
A_5	0	0	0	1	0	1	1	0	0	0	0	0	0
B_1	0	0	0	0	0.126	0	0	0	0	1	0	0	0
B_2	0	0	0	0	0	0	0	0	0	1	1	0	0
B_3	0	0	0	0	0	0	0	0	0	1	1	1	0
B_4	0	0	0	0	0	0	0	0	1	0	1	1	1
D_u	0	0	0	0	0.126	0	0	0	1	1	0	0	1

4.3.4 Build VSM model with semi-supervised PLSA

Following the EM algorithm for semi-supervised PLSA discussed in section 4.1.1.2, we first define the number of maximum iteration R = 500, number of latent topics G = 3, and ε = 1E-5. For the simplicity of problem, we define α = 0.9. The three latent topics are denoted as z_1, z_2 and z_3, respectively. We also need to generate the connection matrix M_{conn} between each document-document pair (D, D') in HCI_GT, where each entry denotes connection value $l(D, D')$ between D and D', as discussed in section 4.2.2. The result is shown in Table 18:

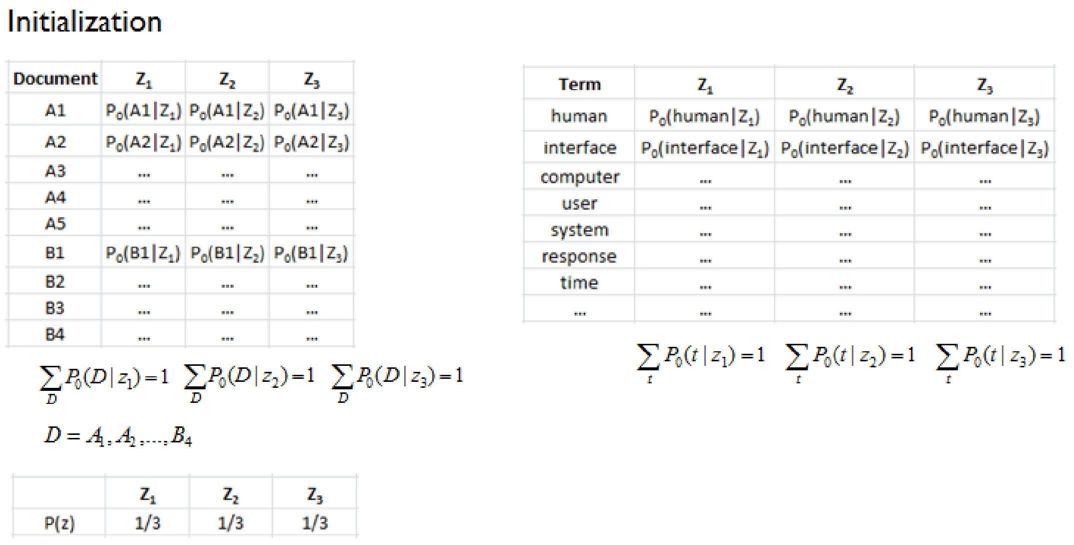
Table 18 Connection matrix for HCI_GF

	***A*₁**	***A*₂**	***A*₃**	***A*₄**	***A*₅**	***B*₁**	***B*₂**	***B*₃**	***B*₄**
***A*₁**									
***A*₂**	1								
***A*₃**	1	1.126							
***A*₄**	1	0.126	1.126						
***A*₅**	1	3	1	0					
***B*₁**	0	0	0	0	0				
***B*₂**	0	0	0	0	0	1.126			
***B*₃**	0	0	0	0	0	1	2		
***B*₄**	0	0	0	0	0	0	1	2	
***D*ᵤ**	0	1.126	0.126	0.126	0	1.126	1	1	1

The steps of building SSTD matrix for training documents in *Tr* are described as following:

- At the initialization step, for each document-topic and topic-word pair, assign random values to $P_0(D \mid z)$ and $P_0(t \mid z)$ between 0 and 1, with the constraints $\sum_D P_0(D \mid z) = 1$, and $\sum_t P_0(t \mid z) = 1$. For each topic, initialize $P_0(z) = \frac{1}{3}$, $z = z_1, z_2, z_3$. As a result, we have the following two matrixes, M_{P_Dz} and M_{P_tz}, as shown in the following Fig. 29.
- At E-step, at iteration *r*, $P_r(z_1 \mid D,t), P_r(z_2 \mid D,t), P_r(z_3 \mid D,t)$ are calculated for each term-document pair using equation (16). Also, $P_r(z_1 \mid D,D'), P_r(z_2 \mid D,D'), P_r(z_3 \mid D,D')$ are calculated for each document-document pair in *Tr* using equation (17), where $D, D' \in Tr, l(D,D') > 0$. The process of E-step in iteration 1 is shown in Fig. 30.
- At M-step, at iteration *r*, for each document-topic and topic-word pair, compute $P_r(D \mid z)$ and $P_r(t \mid z)$ based on the updating formulas in (23) and (24), and compute $P_r(z)$ based on the updating formula in (25). Apparently, this step uses the calculated conditional

posterior probability $P_r(z|D,t)$ and $P_r(z|D,D')$ to update the probability $P_{r-1}(D|z)$, $P_{r-1}(t|z)$ and $P_{r-1}(z)$ into $P_r(D|z)$, $P_r(t|z)$ and $P_r(z)$. The process of M-step in iteration 1 is shown in Fig. 31.



Initialization

Document	Z_1	Z_2	Z_3
A1	$P_0(A1\|Z_1)$	$P_0(A1\|Z_2)$	$P_0(A1\|Z_3)$
A2	$P_0(A2\|Z_1)$	$P_0(A2\|Z_2)$	$P_0(A2\|Z_3)$
A3	...	...	...
A4	...	...	...
A5	...	...	...
B1	$P_0(B1\|Z_1)$	$P_0(B1\|Z_2)$	$P_0(B1\|Z_3)$
B2	...	...	...
B3	...	...	...
B4	...	...	...

$\sum_D P_0(D|z_1)=1$ $\sum_D P_0(D|z_2)=1$ $\sum_D P_0(D|z_3)=1$

$D = A_1, A_2, \ldots, B_4$

Term	Z_1	Z_2	Z_3
human	$P_0(\text{human}\|Z_1)$	$P_0(\text{human}\|Z_2)$	$P_0(\text{human}\|Z_3)$
interface	$P_0(\text{interface}\|Z_1)$	$P_0(\text{interface}\|Z_2)$	$P_0(\text{interface}\|Z_3)$
computer	...	...	...
user	...	...	...
system	...	...	...
response	...	...	...
time	...	...	...
...	...	...	...

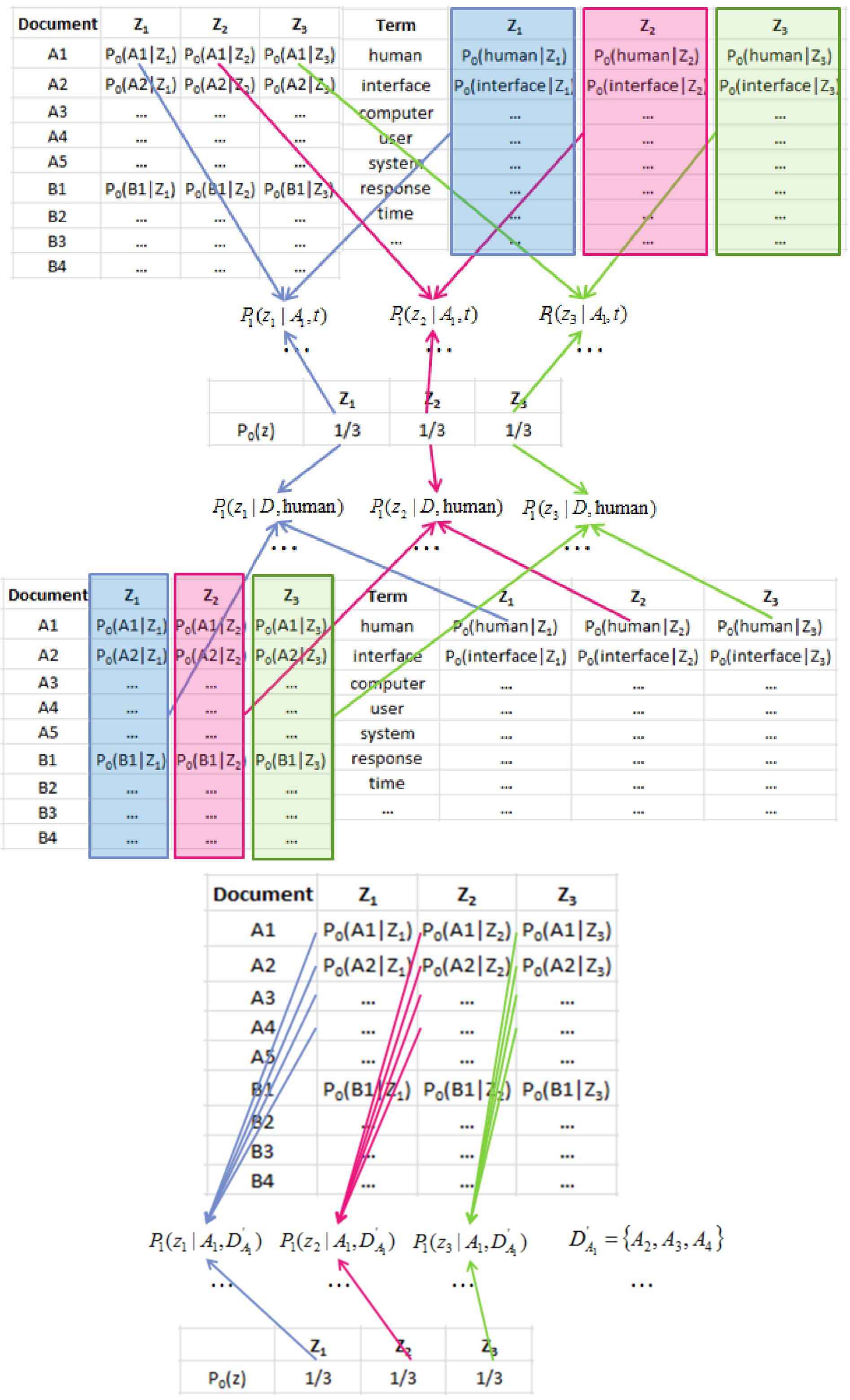
$\sum_t P_0(t|z_1)=1$ $\sum_t P_0(t|z_2)=1$ $\sum_t P_0(t|z_3)=1$

	Z_1	Z_2	Z_3
P(z)	1/3	1/3	1/3

Figure 29. Example of EM algorithm Initialization for semi-supervised PLSA

- The above E-step and M-step repeat until the maximum iteration *R*, or the log-likelihood function $L_r(\theta)$ in (1) met the criterion that $L_r(\theta) - L_{r-1}(\theta) \le \varepsilon$, at iteration *r*. Here, we denote the final estimation for each topic-document pair, each topic-word pair and each latent topic after the above training process as $P_{Tr}(D|z)$, $P_{Tr}(t|z)$ and $P_{Tr}(z), z = z_1, z_2, z_3$.
- We then generate matrix M_{P_zD}, where each entry represents the final conditional probability $P_{Tr}(z|D)$, and $P_{Tr}(z|D) = \dfrac{P_{Tr}(D|z)P_{Tr}(z)}{P(D)} = \dfrac{P_{Tr}(D|z)P_{Tr}(z)}{\sum_z P_{Tr}(D|z)P_{Tr}(z)}$.
- The final results of M_{P_zD} and M_{P_tz} after convergence, with size 9×3 and 13×3 respectively, is shown in Fig. 32.

E-Step : Iteration 1

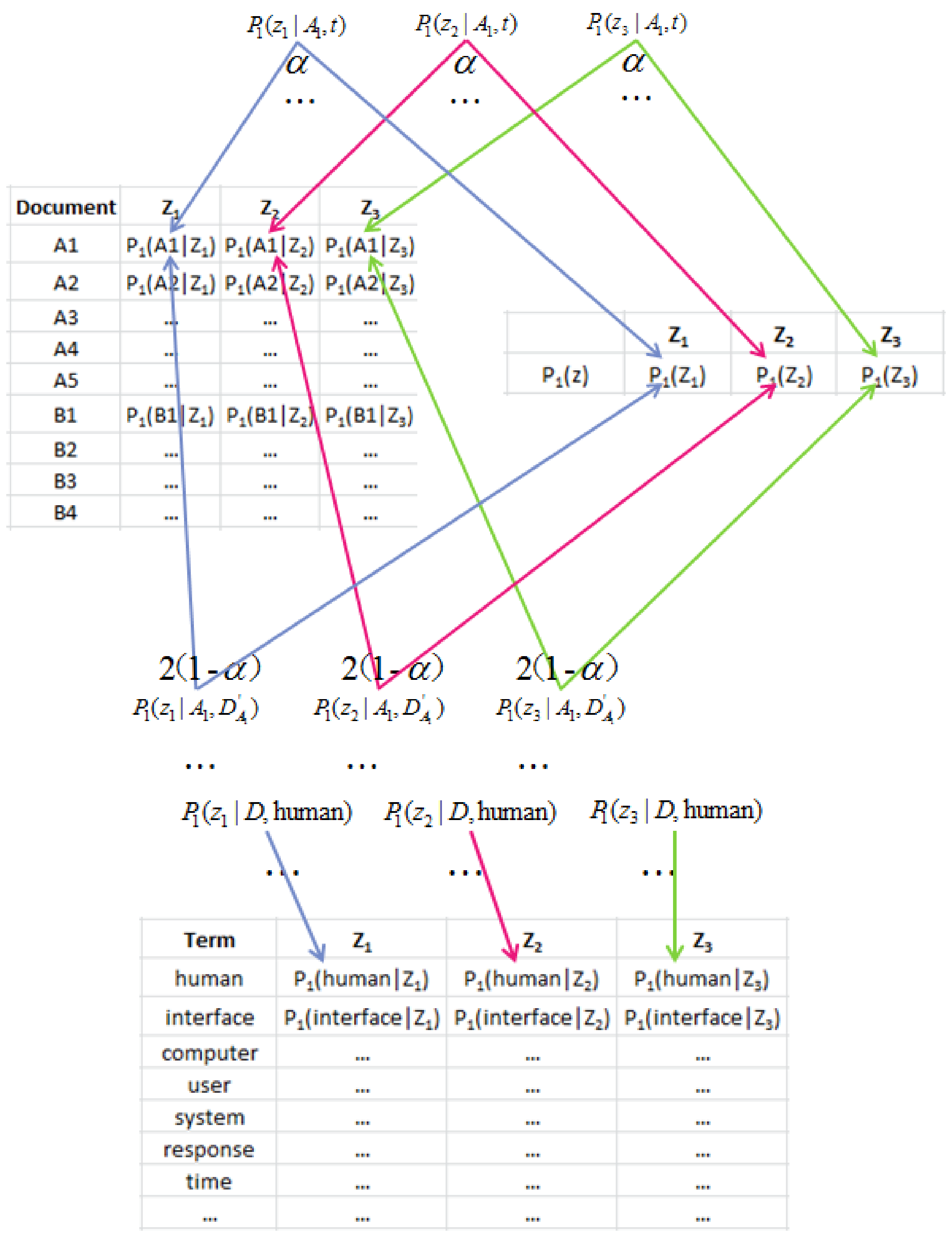

$P_1(z_1 \mid A_1, t)$
$P_1(z_2 \mid A_1, t)$
$P_1(z_3 \mid A_1, t)$
$P_1(z_1 \mid D, \text{human})$
$P_1(z_2 \mid D, \text{human})$
$P_1(z_3 \mid D, \text{human})$
$P_1(z_1 \mid A_1, D_{A_1}')$
$P_1(z_2 \mid A_1, D_{A_1}')$
$P_1(z_3 \mid A_1, D_{A_1}')$
$D_{A_1}' = \{A_2, A_3, A_4\}$
$P_0(z)$
Z_1 Z_2 Z_3
1/3 1/3 1/3

Document	Z_1	Z_2	Z_3
A1	$P_0(A1 \mid Z_1)$	$P_0(A1 \mid Z_2)$	$P_0(A1 \mid Z_3)$
A2	$P_0(A2 \mid Z_1)$	$P_0(A2 \mid Z_2)$	$P_0(A2 \mid Z_3)$
A3	...	...	...
A4	...	...	...
A5	...	...	...
B1	$P_0(B1 \mid Z_1)$	$P_0(B1 \mid Z_2)$	$P_0(B1 \mid Z_3)$
B2	...	...	...
B3	...	...	...
B4	...	...	...

Term	Z_1	Z_2	Z_3
human	$P_0(\text{human} \mid Z_1)$	$P_0(\text{human} \mid Z_2)$	$P_0(\text{human} \mid Z_3)$
interface	$P_0(\text{interface} \mid Z_1)$	$P_0(\text{interface} \mid Z_2)$	$P_0(\text{interface} \mid Z_3)$
computer	...	...	...
user	...	...	...
system	...	...	...
response	...	...	...
time	...	...	...
...	...	...	...

Figure 30. Example of EM algorithm E-Step for semi-supervised PLSA

M-Step: Iteration I



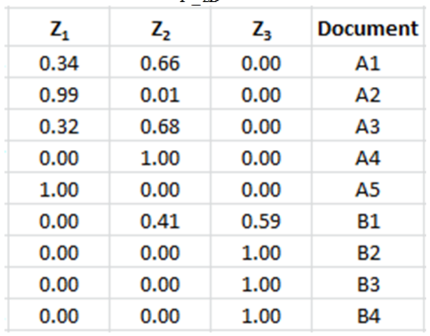
Document	Z_1	Z_2	Z_3
A1	$P_1(A1\|Z_1)$	$P_1(A1\|Z_2)$	$P_1(A1\|Z_3)$
A2	$P_1(A2\|Z_1)$	$P_1(A2\|Z_2)$	$P_1(A2\|Z_3)$
A3	...	...	...
A4	...	...	...
A5	...	...	...
B1	$P_1(B1\|Z_1)$	$P_1(B1\|Z_2)$	$P_1(B1\|Z_3)$
B2	...	...	...
B3	...	...	...
B4	...	...	...

	Z_1	Z_2	Z_3
$P_1(z)$	$P_1(Z_1)$	$P_1(Z_2)$	$P_1(Z_3)$

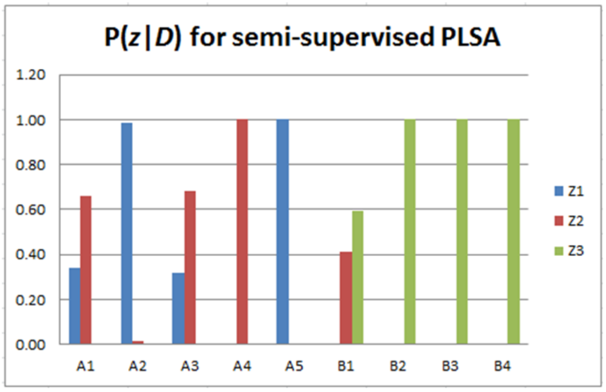
Term	Z_1	Z_2	Z_3
human	$P_1(\text{human}\|Z_1)$	$P_1(\text{human}\|Z_2)$	$P_1(\text{human}\|Z_3)$
interface	$P_1(\text{interface}\|Z_1)$	$P_1(\text{interface}\|Z_2)$	$P_1(\text{interface}\|Z_3)$
computer	...	...	...
user	...	...	...
system	...	...	...
response	...	...	...
time	...	...	...
...	...	...	...

Figure 31. Example of EM algorithm M-Step for semi-supervised PLSA

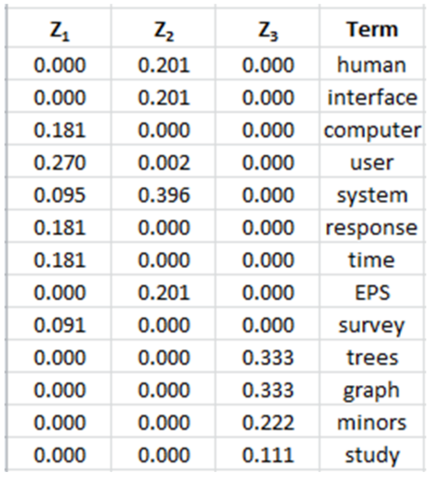
M_{P_zD}



Z_1	Z_2	Z_3	Document
0.34	0.66	0.00	A1
0.99	0.01	0.00	A2
0.32	0.68	0.00	A3
0.00	1.00	0.00	A4
1.00	0.00	0.00	A5
0.00	0.41	0.59	B1
0.00	0.00	1.00	B2
0.00	0.00	1.00	B3
0.00	0.00	1.00	B4



M_{P_tz}



Z_1	Z_2	Z_3	Term
0.000	0.201	0.000	human
0.000	0.201	0.000	interface
0.181	0.000	0.000	computer
0.270	0.002	0.000	user
0.095	0.396	0.000	system
0.181	0.000	0.000	response
0.181	0.000	0.000	time
0.000	0.201	0.000	EPS
0.091	0.000	0.000	survey
0.000	0.000	0.333	trees
0.000	0.000	0.333	graph
0.000	0.000	0.222	minors
0.000	0.000	0.111	study

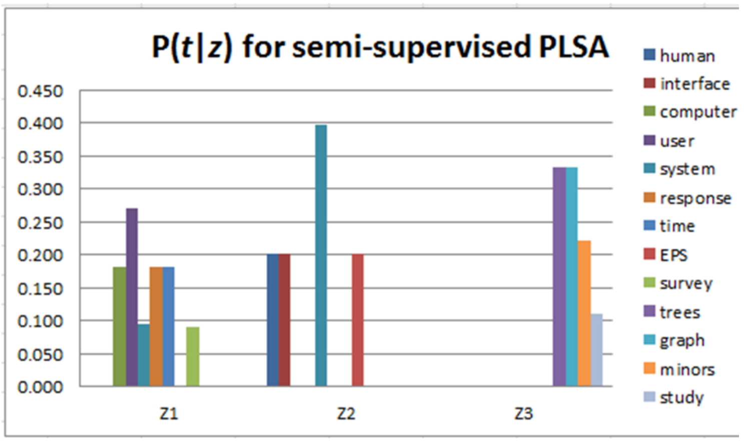


Figure 32. Example of EM algorithm result for semi-supervised PLSA after convergence

In M_{P_zD}, comparing with Fig. 27 for PLSA, we observed an increase on the probability of latent topic z_1 given document A_1 and A_3. This is because that A_1 is connected to A_2 by term "computer", and A_3 is connected to A_2 by term "user". These two terms all have probability of occurrence on z_1, which contributes to the increase of $P(z_1 \mid A_1)$ and $P(z_1 \mid A_3)$. Also, the probability of latent topic z_3 given document B_1 increased, because that B_1 only connects to B_2 and B_3 by term "tree", which only occurs in latent topic z_3.

For testing document D_u, we run through the EM algorithm to generate the conditional probability of each latent topic z for D_u using the following procedure with the same parameter setting for R, G and ε:

- At the initialization step, for each topic given D_u, assign random values to $P_0(z \mid D_u)$ between 0 and 1, with the constraints $\sum_{z=z_1}^{z_3} P_0(z \mid D_u) = 1$. Then $P_0(D_u \mid z)$ is calculated using the following formula: $P_0(D_u \mid z) = \dfrac{P_0(z \mid D_u)\dfrac{1}{P_{Tr}(z)}}{\sum_z P_0(z \mid D_u)\dfrac{1}{P_{Tr}(z)}}$, $z = z_1, z_2, z_3$.

  $P(D_u)P_0(D_u \mid z) = P_0(z \mid D_u)\dfrac{1}{P_{Tr}(z)}$,

- At E-step, at iteration *r*, $P_r(z_1 \mid D_u, t), P_r(z_2 \mid D_u, t)$ and $P_r(z_3 \mid D_u, t)$ are calculated for each term given D_u using equation (16), which use $P_{Tr}(t \mid z)$ and $P_{Tr}(z)$ generated from the training process. Also, $P_r(z_1 \mid D_u, D_u^{'}), P_r(z_2 \mid D_u, D_u^{'})$ and $P_r(z_3 \mid D_u, D_u^{'})$ are calculated for each pair of $(D_u, D_u^{'})$ using equation (17), where $D_u^{'} \in Tr, l(D_u, D_u^{'}) > 0$, which use $P_{Tr}(D_u^{'} \mid z)$ and $P_{Tr}(z)$ generated from the training process.
- At M-step, at iteration *r*, compute $P_r(D_u \mid z)$ based on the updating formulas in (24).
- The above E-step and M-step repeat until the maximum iteration *R*, or the log-likelihood function $L_r(\theta)$ in (1) met the criterion that $L_r(\theta) - L_{r-1}(\theta) \le \varepsilon$, at iteration *r*. Here, we denote the final estimation for D_u given latent topic *z* as $P_F(D_u \mid z)$.
- The final conditional probability $P_F(z \mid D_u)$ for each topic given D_u can be calculated using: $P_F(z \mid D_u) = \dfrac{P_F(D_u \mid z)P_{Tr}(z)}{P(D_u)} = \dfrac{P_F(D_u \mid z)P_{Tr}(z)}{\sum_{\mathrm{z}} P_F(D_u \mid z)P_{Tr}(z)}$.

Therefore, for testing document D_u, the final topic-document representation is an 1×3 vector V_{D_u}, where $V_{D_u} = [P_F(z_1 \mid D_u), P_F(z_2 \mid D_u), P_F(z_3 \mid D_u)]$, as shown in Fig. 33.

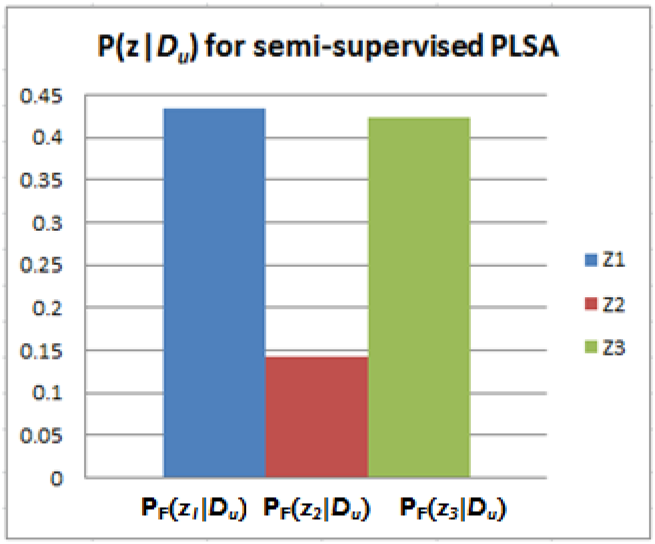


Figure 33. Semi-supervised PLSA EM algorithm result for testing document D_u

Comparing with Fig. 28 for PLSA, the probability of latent topic z_3 increases, while the probability of latent topic z_2 decreases, given testing document D_u. This is due to the fact that, when we take a look at $P_{Tr}(D \mid z)$ generated for training documents in Fig. 34 as well as the connection matrix in Table 18, D_u is connected to B_2, B_3 and B_4 that have a high probability on z_3, and is connected to A_2 that has a high probability on z_1, with high connection values. Also, although D_u is connected to A_3 and A_4 that have high probability on z_2, the term "system" has a low weight in TD matrix, which contributes to the low value of $l(D_u, A_3)$ and $l(D_u, A_4)$. These above factors all contributes to the final estimation of $P_F(z \mid D_u)$, where $P_F(z_2 \mid D_u)$ is much lower than $P_F(z_1 \mid D_u)$ and $P_F(z_3 \mid D_u)$.

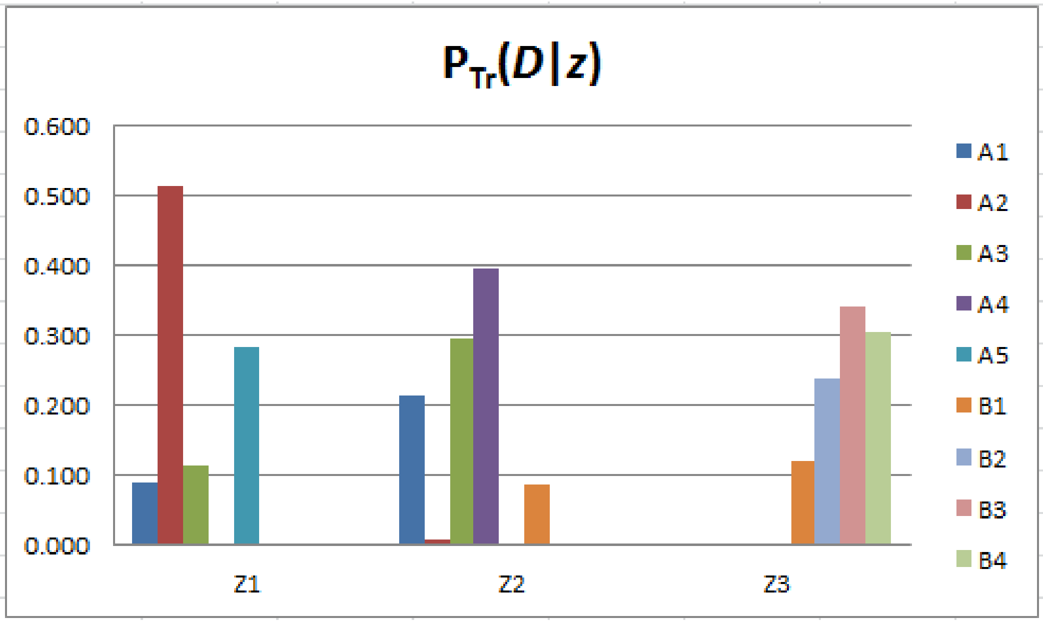


Figure 34. $P_{Tr}(D|z)$ generated for semi-supervised PLSA

We can see that instead of only considering word co-occurrence information in PLSA, the semi-supervised PLSA approach also incorporates document connectivity information extracted from both semantic information provided by WordNet and document category labels, thus provides more reasonable topic-document features than PLSA. An overview of PLSA and semi-supervised PLSA comparison is presented in the following Fig. 35.

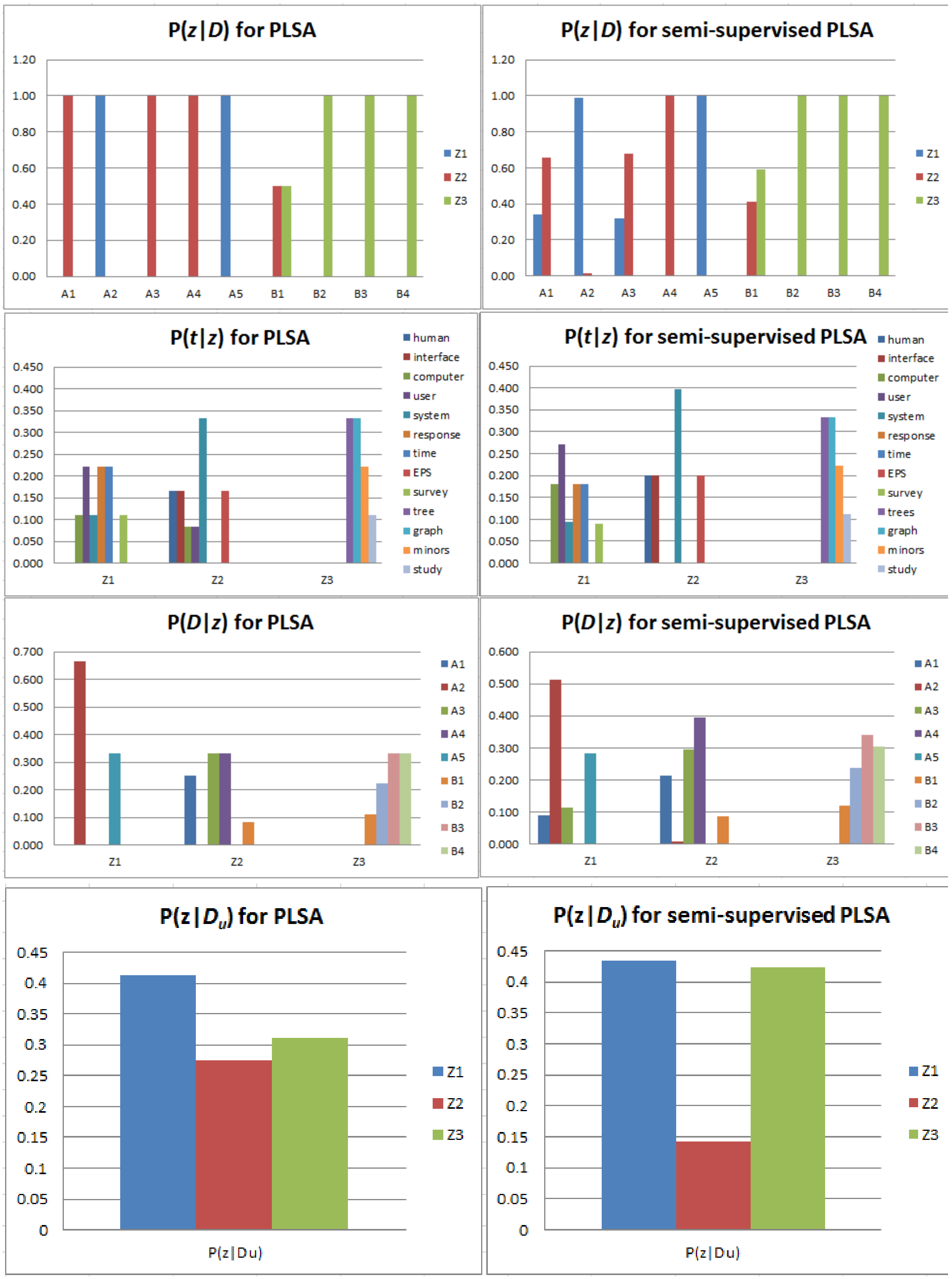


Figure 35. Comparison between PLSA and semi-supervised PLSA

4.3.5 System evaluation based on document distance measure

In order to evaluate each procedure during VSM generation, we calculate the Euclidean distance of the vector representation between testing document D_u and each of the 9 training documents in *Tr*, as shown in the following Table 19 and 20.

Table 19 Euclidean distance between D_u and training documents based on different text representation - I

Document	Content	Euclidean distance to D_u		
		tf	*tf - idf*	*tf - CE_W*
$\boldsymbol{A_1}$	***Human*** *machine* ***Interface*** *for ABC* ***computer*** *applications*	2.449	1.414	2.240
$\boldsymbol{A_2}$	*A* ***survey*** *of* ***user*** *opinion of* ***computer system response time***	2.236	0.919	2.236
$\boldsymbol{A_3}$	*The* ***EPS user interface*** *management* ***system***	2.236	1.341	2.236
$\boldsymbol{A_4}$	***System*** *and* ***human system*** *engineering testing of* ***EPS***	2.236	1.288	2.002
$\boldsymbol{A_5}$	*Relation of* ***user*** *perceived* ***response time*** *to error management*	2.449	1.414	2.240
$\boldsymbol{B_1}$	***System*** *of random, binary, ordered* ***tree***	1.000	0.972	1.000
$\boldsymbol{B_2}$	*The intersection* ***graph*** *of paths in* ***tree***	1.732	1.183	1.420
$\boldsymbol{B_3}$	***Graph minors*** *IV: Widths of* ***tree*** *and well-quasi-ordering*	2.000	1.252	1.737
$\boldsymbol{B_4}$	***Graph minors****: A* ***study***	2.449	1.414	2.240
$\boldsymbol{D_u}$	*A* ***survey*** *of decision* ***tree system***			

From Table 19, the effect of global weighting scheme is obviously significant. With only *tf* representation, it cannot differentiate A_2, A_3 and A_4, as well as A_1, A_5 and B_4. With *tf-idf* representation, A_2, A_3 and A_4 are differentiated, with A_2 identified as the closest document to D_u, because that two terms, "survey" and "system" occurs in both A_2 and D_u, which is not a very good assignment. With *tf-CE_W* representation, the closest document to D_u is changed to B_1, because of the low weight assigned to "system", which is more reasonable. However, A_2 and A_3, as well as A_1, A_5 and B_4 still cannot be differentiated.

Table 20 Euclidean distance between D_u and training documents based on different text representation - II

Document	Content	Euclidean distance to D_u			
		tf-idf	*tf-idf +PLSA*	*tf-CE_W +PLSA +WordNet*	*tf-CE_W +semi-supervised PLSA +WordNet*
A_1	***Human** machine **Interface** for ABC **computer** applications*	1.414	1.671	2.794	2.734
A_2	*A **survey** of **user** opinion of **computer system response time***	0.919	1.166	2.349	2.346
A_3	*The **EPS user interface** management **system***	1.341	1.610	2.791	2.735
A_4	***System** and **human system** engineering testing of **EPS***	1.288	1.566	2.608	2.667
A_5	*Relation of **user** perceived **response time** to error management*	1.414	1.586	2.744	2.745
B_1	***System** of random, binary, ordered **tree***	0.972	1.096	1.805	1.814
B_2	*The intersection **graph** of paths in **tree***	1.183	1.457	2.177	2.135
B_3	***Graph minors** IV: Widths of **tree** and well-quasi-ordering*	1.252	1.514	2.396	2.358
B_4	***Graph minors**: A **study***	1.414	1.650	1.934	1.886
D_u	*A **survey** of decision **tree system***				

From Table 20, we can see the significant effect of PLSA and WordNet ontology. With *tf-idf* representation, plus the three latent topic features generated from PLSA, similar to the effect of applying *CE_W*, the closest document to D_u is changed to B_1, because of their similar topic probability distribution (both have probability on z_2 and z_3). Also A_2 and A_3, as well as A_1, A_5 and B_4 are differentiated from each other, based on the probability of occurring terms in these documents on the three latent topics.

By adding CD matrix and modifying TD matrix using WordNet, apparently, A_2 and B_4 are ranked closer to D_u, due to the semantic similarity between "survey" and "study". After this stage, we have a strong confidence that D_u has a higher probability of belonging to category B.

From the final column of Table 20, we can see that the semi-supervised PLSA has the similar effect as PLSA, showing that this approach is as robust as PLSA. Moreover, A_1 and A_3 are ranked closer to D_u than A_5, because that connectivity among A category documents increase their probability on latent topic z_1. Distance from D_u to A_4 increases because of the probability decrease on $P_F(z_2 \mid D_u)$, and distances from D_u to B_2 , B_3 and B_4 decreases because of the probability increase on $P_F(z_3 \mid D_u)$. Therefore, we may conclude that semi-supervised PLSA provides a more reasonable text representation by generating semi-supervised topic-document features based on semantic relationship between words and connectivity between documents.

4.4 Generate hybird VSM model for classification

Considering the task of text categorization, the procedures discussed in section 4.1 and 4.2 are combined together to generate a hybrid VSM text representation. For training set *Tr*, the process of VSM matrix generation and combination in our final system, using WordNet ontology and semi-supervised PLSA, is shown in the following Fig. 36. Starting from TD frequency matrix, our system generates TD matrix M_0 with global weighting scheme. TD matrix M_0 is then used to generate concept-document matrix M_c using WordNet ontology, and M_0 is also replaced by M_1 based on word relationships extracted from WordNet ontology. The topic-document matrix M_{Ltd} or M_{sstd} using PLSA / semi-supervised PLSA modeling is then generated based on M_0 and M_c. Note here, the feature size of the final matrix is the sum of original indexed terms, number of concepts generated from *Tr* and number of latent topics we defined. These features could be further selected using feature selection techniques such as Gini Index, Information Gain, Mutual Information, etc [65,66]. However, to make the text representation features as inclusive as possible, we still keep all of them. The hybrid text representation model is then used to learn and evaluate classifiers, e.g., SVM, Neural Networks, Naïve Bayes classifier, etc.

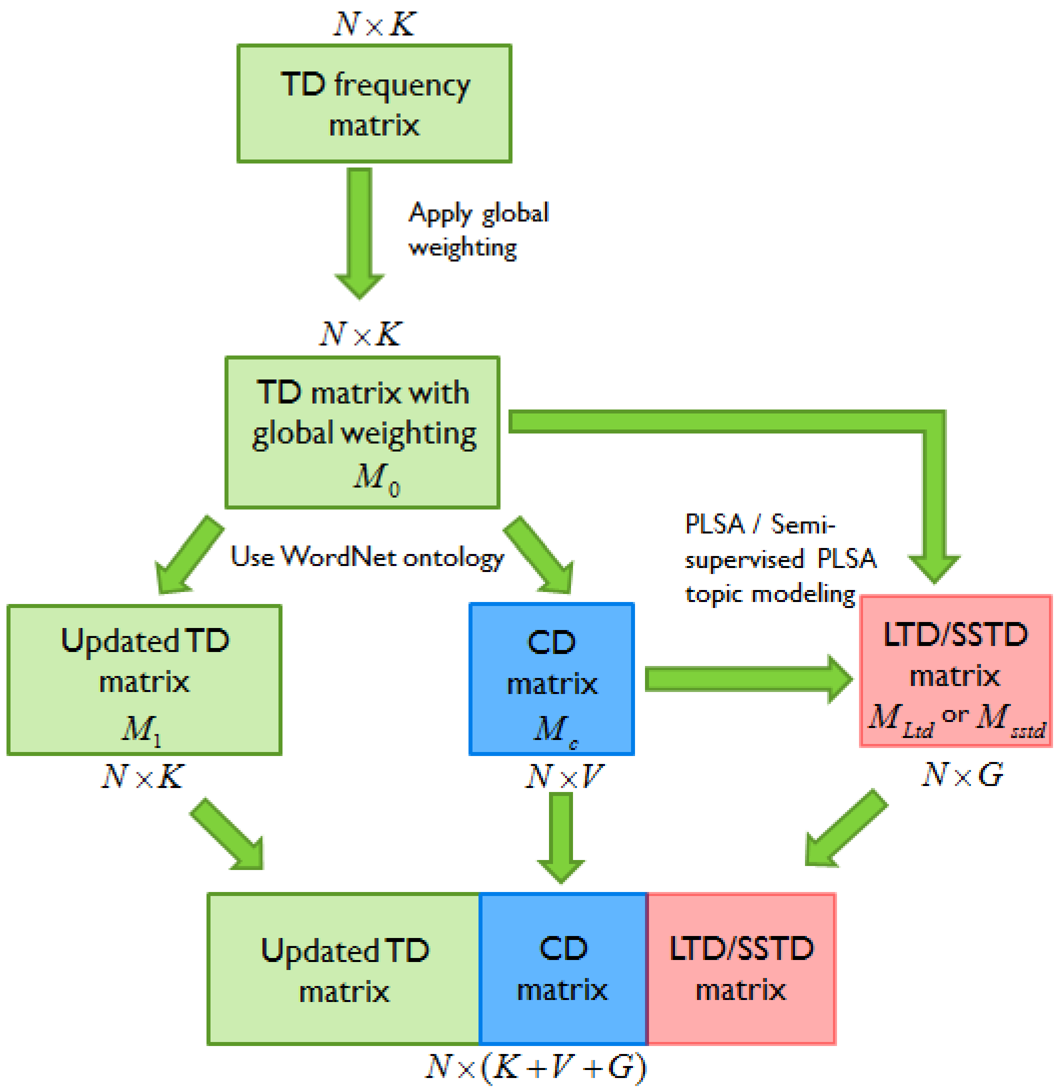


Figure 36. VSM matrix generation and combination

CHAPTER 5. EMPIRICAL CASE STUDY AND EXPERIMENTAL RESULTS EVALUATION ON TEXT CATEGORIZATION

In this section, we present experiments we conducted using our proposed text categorization framework in Fig. 18 and Fig. 20, and classification results analysis on several publicly available or domain-specific datasets. This is an extension of experiments on HCI_GT discussed in section 4.3, in terms of investigating how the proposed text representation could help improving text categorization accuracy. The experiments are designed as comparisons of the proposed text representation with conventional text representation methods. We present the experiment results of tuning parameters for PLSA, semi-supervised PLSA and WordNet ontology, and provide a detailed performance analysis of our proposed text representation approach.

5.1 Datasets

In this empirical study, we first use three publicly available and widely used datasets to evaluate our proposed system. These datasets include Reuters-21578 [115], Nist Topic Detection and Tracking corpus (TDT2) [114], and 20 newsgroups [116]. Reuters-21578 corpus contains 21578 documents in 135 categories. After removing documents with multiple category labels, it left 8,293 documents in 65 categories. In TDT2, those documents appearing in two or more categories were removed, and only the largest 30 categories were kept, thus leaving 9,394 documents in total. 20 newsgroups dataset is a collection of 18,846 newsgroup documents, partitioned (nearly) evenly across 20 different newsgroups.

To evaluate the system performance on domain-specific datasets that has customized category definition such as [24,80], we also used a dataset named VDR, that contains 600 vehicle diagnostic records, in which documents that contain descriptions that reveal systematic engineering or manufacturing failures are defined as of interests (Category-A), and all other documents belong to Category-B. The major challenge in this problem is that the documents of interests are not explicitly defined by either topics or general descriptions, as shown in the following examples:

Category-A document: "perform abs self roadtest found rear wheel speeds sensor connector corroded into sensor replace sensor and connector road tester ok clear code"

Category-B document: "road roadtest traction control lamp on eec roadtest code c1280 u415 om rcm contact hot line 103912699 check connection at rcm check mounting bolts ok clear code"

In all of these datasets discussed above, preprocessing tasks mentioned in section 4.1.1 are conducted and stop words are removed. Note here, all words having occurrence frequency lower than $\tau = 5$ are removed, except VDR dataset. TD matrix weighted by *CE_W* is then generated for each dataset. TD matrix based on *tf* only is also generated for PLSA model learning, and TD matrix weighted by *idf* is generated for evaluation purpose.

5.2 Experiment setup

Considering that the focus of this work is not improving or comparing machine learning algorithms, we use SVM as our classification model throughout different experiments. SVM training is carried out with LIBSVM package, which is developed by Chih-Chung Chang and Chih-Jen Lin from National Taiwan University [117]. For each dataset, we did 3-fold cross validation, and in each fold, we choose 2/3 documents from each class as training set, and the remaining 1/3 documents as testing set. We apply the Gaussian Radial Basis kernel function (RBF) and tune the parameter gamma to 0.001, 0.001, 0.1 and 0.1, for Reuters, TDT2, 20newsgroups and VDR, respectively, based on the average testing accuracy of the 3 folds.

All experiments are performed on a desktop with Intel(R) Core i7 processor operating at 3.40GHz and 16 GB of memory, with 64-bit Windows 7 system, JDK7.0 + Netbeans 7.3.1, and Matlab 2009a.

5.3 *Build VSM model with PLSA*

Similar to the example discussed in section 4.3.2, in PLSA learning, for all datasets, we define the number of maximum iteration R = 500 for training set, and R = 200 for testing set. The convergence goal is defined as ε = 1E-5. An example of log-likelihood function maximization on Reuters dataset is shown in Figure 37. It is obvious that the log-likelihood function converges very fast and become very stable after 300 iterations.

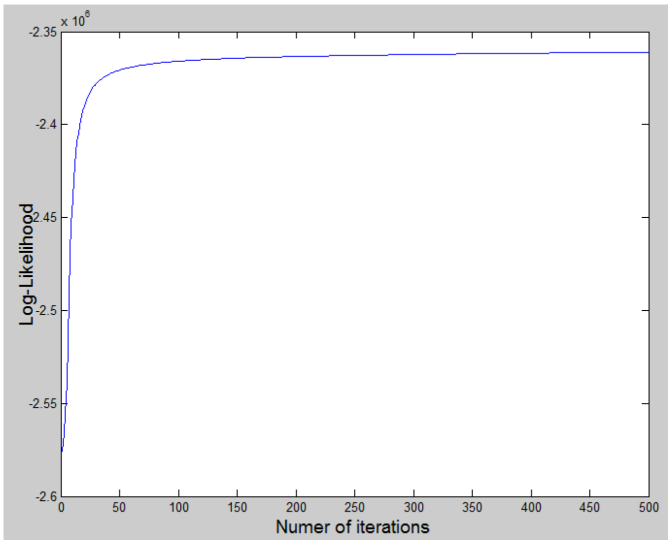


Figure 37. Example of log-likelihood maximization for PLSA

We performed experiments on text categorization by investigating what topic number is the most appropriate for different datasets, using topic-document features generated by PLSA as text representation. The results are illustrated as following in Table 21 and Fig. 38. From the results, we observe that for Reuters, TDT2, 20news and VDR dataset, 60 topics, 140 topics, 140 topics and 30 topics yields the best categorization accuracy, respectively. As a result, generally we should define larger number of topics during PLSA modeling on dataset with larger number of terms, but the relationship between term size and number of topics is not simply monotonic and linear. In order to obtain promising categorization performance, selecting an appropriate number of topics that could best differentiate documents in different classes is very important. In practice, we may conclude that it is reasonable to set the number of topics around $\sqrt{K}$, where K is the number of indexed terms. For the convenience of evaluation and analysis, in the later experiments, we keep on using 60 topics for Reuters, 140 topics for TDT2 and 20news dataset, and 30 topics for VDR dataset.

Table 21 Text categorization performance based on different number of topics generated by PLSA

LTD matrix generated by PLSA	# of Topic = 20	# of Topic = 30	# of Topic = 40	# of Topic = 60	# of Topic = 80	# of Topic = 100	# of Topic = 120	# of Topic = 140	# of Topic = 160	# of Topic = 180
Reuters (8558 terms)	81.27%	80.91%	81.42%	**84.60%**	83.09%	83.33%	82.87%	81.43%	81.55%	80.12%
TDT2 (19448 terms)	79.34%	84.39%	85.01%	87.01%	88.62%	89.28%	88.38%	**90.24%**	88.23%	89.97%
20news (24347 terms)	63.28%	63.67%	67.12%	74.24%	78.51%	78.07%	77.33%	**79.46%**	76.90%	72.13%
VDR (1062 terms)	76.79%	**82.32%**	81.21%	80.11%	77.34%	74.03%	72.87%	70.25%	71.22%	69.55%

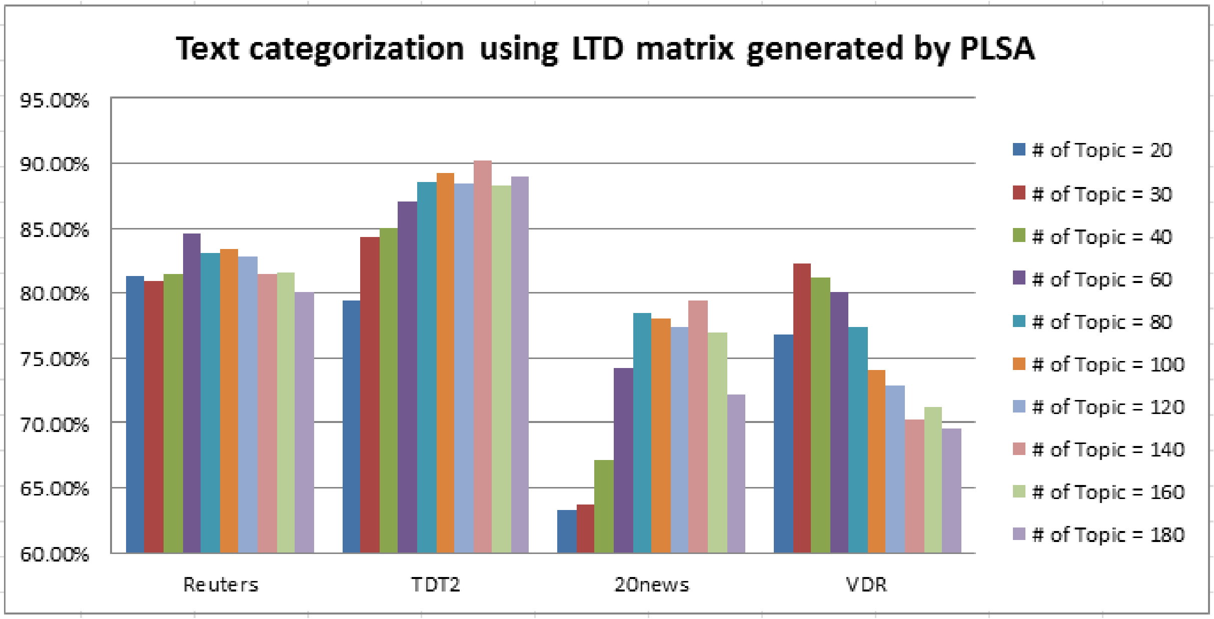


Figure 38. Text categorization performance based on different number of topics generated by PLSA

5.4 Build VSM model with WordNet ontology

Similar to the example in section 4.3.3, WordNet ontology network is utilized in our text categorization model using "add" rule to generate CD matrix, and "replace" rule to replace weighted TD matrix. In our experiments, we first looked into the effect of text categorization using terms within different word class, as discussed in section 4.2.1. The results are shown in Table 22. It is obvious that the best word class is "Noun", which generates 377, 1161, 621 and 21 "concept" features for Reuters, TDT2, 20newsgroups and VDR, respectively, and having a promising text categorization accuracy. Although mixed word class also has the similar accuracy as "Noun", it generates much higher dimensionality of feature space. Therefore, for the rest of our experiments, while performing WordNet synonym searching, we consistently only consider "Noun" synset for each indexed term in *T_L* generated from training set *Tr*.

Secondly, in order to investigate the effect of hypernym/hyponym and meronym/holonym relationships in generating "concept" features, we define the weights of the edges for hypernym/hyponym and meronym/holonym relationships, μ and ν, as $\mu \in \{0, 0.25, 0.5, 0.75, 1\}$, and $\nu = \frac{\mu}{2}$. The results of text categorization accuracy using different weight values for hypernym/hyponym and meronym/holonym relationships are shown in Table 23. From the results, for Reuters and 20news datasets, $\mu = 0.25$ yields the best performance, while TDT2 and VDR datasets have highest accuracy when $\mu = 0.5$. Therefore, we may conclude that hypernym/hyponym and meronym/holonym relationships do help with text categorization, but should not be assigned with a too high weight (e.g., less than 0.5), since synonymy is the most important semantic relationship in generating "concept" based features.

Table 22 Text categorization performance using WordNet based on different word class

TD matrix with *tf-CE_W* + CD matrix ($\mu = 0.25$)				
Categorization accuracy	Noun	Verb	Adjective	Mixed
Reuters	**92.74%**	92.34%	89.01%	92.26%
TDT2	**97.07%**	96.22%	96.15%	96.64%
20news	**87.68%**	86.48%	86.58%	86.73%
VDR	**84.53%**	82.32%	82.32%	83.97%
# of concepts generated	Noun	Verb	Adjective	Mixed
Reuters	377	229	138	677
TDT2	1161	666	645	2201
20news	621	537	172	1165
VDR	21	28	11	58

Table 23 Text categorization performance using WordNet based on different hypernym/hyponym and meronym/holonym weighting

TD matrix with *tf-CE_W* + CD matrix					
Categorization accuracy	$\mu = 0$	$\mu = 0.25$	$\mu = 0.5$	$\mu = 0.75$	$\mu = 1$
Reuters	92.06%	**92.74%**	91.43%	91.31%	90.91%
TDT2	97.07%	97.07%	**98.02%**	97.95%	97.95%
20news	86.61%	**87.68%**	86.56%	85.78%	85.53%
VDR	84.53%	84.53%	**85.63%**	83.43%	83.43%

5.5 Build VSM model with semi-supervised PLSA

Similar to section 5.3, in semi-supervised PLSA learning, for all datasets, we also define the number of maximum iteration R = 500 for training set, and R = 200 for testing set. Number of latent topics is defined as $G = 60$ for Reuters and TDT2, $G = 140$ for 20news dataset, and $G = 30$ for VDR dataset, which is selected based on results from PLSA learning in section 5.3. Also, the convergence goal is defined as ε = 1E-5. To investigate the effect of hyper-weight α that balance the affection of document content and connectivity, we define $\alpha \in \{0.9, 0.8, ...0.1\}$, and the categorization accuracy using latent-topic features generated by semi-supervised PLSA is evaluated by iterating through different values of α.

The accuracy and F-1 measure over different α on all of our experiment datasets are shown in the following Table 24, Table 25, Fig. 39 and Fig. 40. From the result, we could see that, for Reuters, TDT2 and VDR, $\alpha = 0.6$ yields the best text categorization performance. For 20News dataset, $\alpha = 0.8$ yields the best performance, and performance decrease significantly after $\alpha \leq 0.6$, which indicates that in 20News dataset, co-occurrence relationship between words is much more important than document connectivity relationship. As a result, we may conclude that we should not assign too much weight on document connectivity. E.g., $0.6 \leq \alpha \leq 0.8$ is a reasonable range for parameter tuning.

Table 24 Text categorization accuracy using semi-supervised PLSA based on different hyper-weight values

Semi-supervised PLSA				
Categorization accuracy	Reuters	TDT2	20news	VDR
$\alpha = 0.9$	85.55%	89.36%	78.99%	83.42%
$\alpha = 0.8$	82.38%	88.45%	**79.90%**	81.21%
$\alpha = 0.7$	85.59%	79.23%	78.94%	84.53%
$\alpha = 0.6$	**86.15%**	**91.59%**	75.54%	**86.74%**
$\alpha = 0.5$	85.91%	90.71%	75.05%	81.76%
$\alpha = 0.4$	85.56%	88.69%	75.27%	80.66%
$\alpha = 0.3$	85.87%	84.67%	70.41%	77.35%
$\alpha = 0.2$	85.11%	88.31%	62.14%	83.42%
$\alpha = 0.1$	79.17%	88.42%	60.15%	82.32%

Table 25 Text categorization accuracy using semi-supervised PLSA based on different hyper-weight values

Semi-supervised PLSA				
Average F-1 measure	Reuters	TDT2	20news	VDR
$\alpha = 0.9$	0.8077	0.8392	0.7693	0.7842
$\alpha = 0.8$	0.7621	0.8485	**0.7900**	0.7694
$\alpha = 0.7$	0.8261	0.7998	0.7817	0.8235
$\alpha = 0.6$	**0.8272**	**0.8820**	0.7530	**0.8717**
$\alpha = 0.5$	0.8265	0.8542	0.7629	0.8342
$\alpha = 0.4$	0.8256	0.8675	0.7774	0.8222
$\alpha = 0.3$	0.8266	0.8259	0.7037	0.7748
$\alpha = 0.2$	0.8230	0.8468	0.6336	0.7769
$\alpha = 0.1$	0.7987	0.8568	0.6222	0.7434

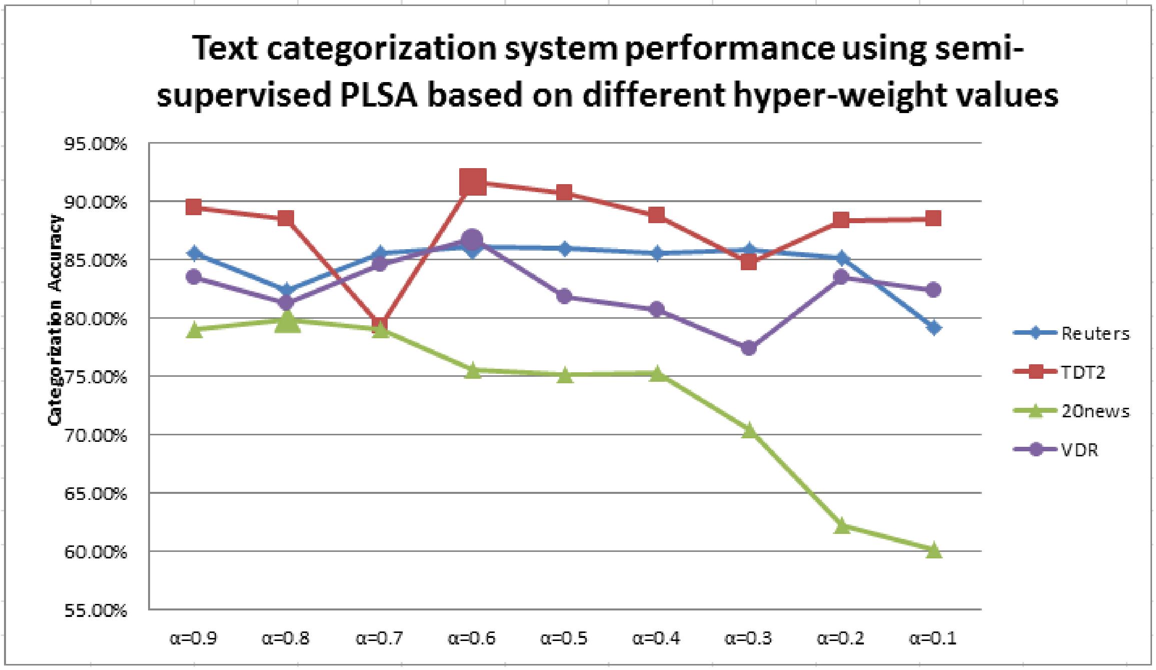


Figure 39. Text categorization performance based on different number of topics generated by PLSA

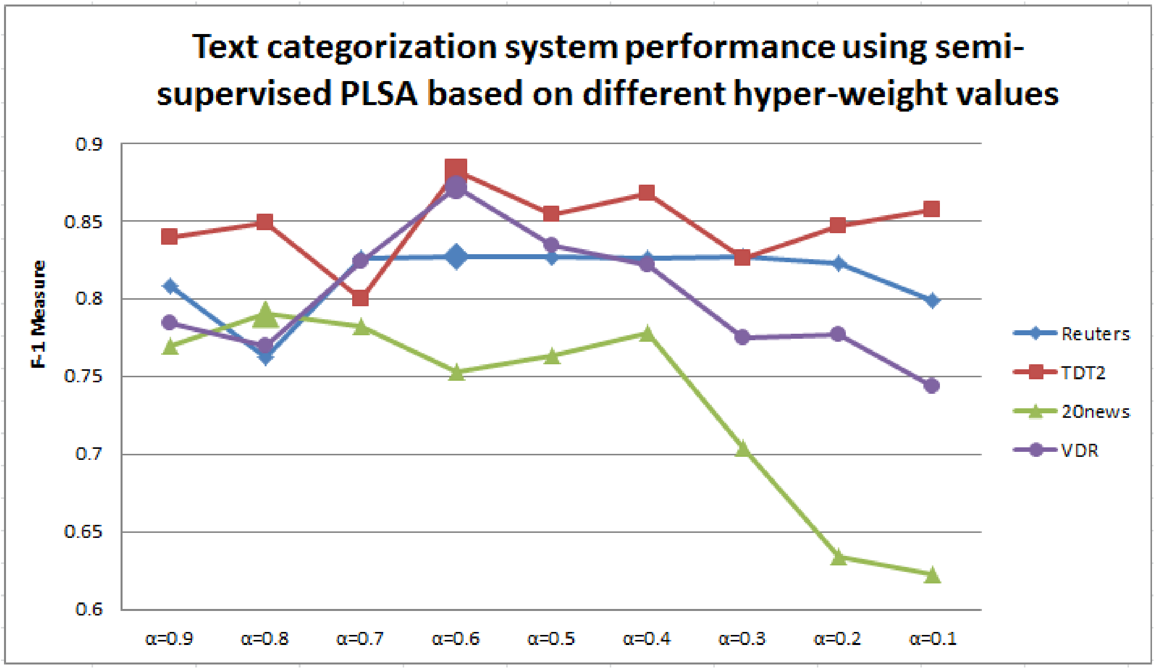


Figure 40. Text categorization performance based on different number of topics generated by PLSA

5.6 Text categorization performance summary & analysis

The classification results are presented in the following Table 26 and 27. We evaluate several systems as our baseline, including TD matrix weighted by *idf*, and using only PLSA generated LTD matrix. From the result, it is obvious that global weighting scheme *CE_W* outperform the *idf* weighting, and the CD matrix generated by WordNet improves categorization accuracy by combining with the weighted TD matrix. The proposed system with *CE_W* weighted and WordNet updated TD matrix, plus CD matrix generated by WordNet and LTD matrix generated by PLSA, significantly outperforms baseline systems, indicating that adding both word relationships and latent semantic information could improve text representation, and the final system with semi-supervised PLSA performs even better. From the result it is interesting to see that, for dataset with specific user-defined groups and having no explicitly defined classification rule as discussed in section 1.2, such as VDR, the latent semantic topic based text representation could already do a better job than *tf-idf* approach. This indicates that for those applications, semantic structure is much more important than word occurrence information. Note here, the performance of our system is also comparable to or outperforms state-of-art approaches, such as those proposed in [121,122,123,124], where the best accuracy on 20News, Reuters and TDT2 dataset are 88.89%, 92.5% and 93.85%, respectively.

Considering the efficiency of our text categorization system, VSM_WN_TM, the additional processing time comparing with traditional VSM approach includes the following aspects:

- Additional features added to the original TD matrix. Considering the original term features are generally 20-40 times larger than added features including concept and latent topics, the additional processing time for classifier training is around 1%.
- Updating TD matrix using WordNet. This step only increase around 0.5% of processing time.
- PLSA model learning for training set and testing document. The processing time of learning process varies on different size of datasets, and the longest training time, which is for 20news dataset, takes around 1 hour. However, this could be further reduced significantly by term feature selection and parameter tuning.

Table 26 Text categorization accuracy comparison

	Reuters	**TDT2**	**20news**	**VDR**
TD matrix weighted by *idf* (Baseline)	91.03%	89.37%	85.85%	80.95%
TD matrix weighted by *CE_W*	92.10%	96.01%	87.25%	85.08%
TD matrix weighted by *idf* + CD matrix	91.46%	95.09%	87.10%	83.07%
TD matrix weighted by *CE_W* + CD matrix (VSM_WN)	**92.74%**	**98.02%**	**87.68%**	**85.63%**
LTD matrix by PLSA	84.60%	90.24%	79.46%	82.32%
SSTD matrix by semi-supervised PLSA (VSM_TM)	86.15%	91.59%	79.90%	86.74%
WordNet-augmented TD matrix weighted by *CE_W* + CD matrix + LTD matrix	**93.06%**	**98.78%**	**88.84%**	**87.84%**
WordNet-augmented TD matrix weighted by *CE_W* + CD matrix +semi-supervised PLSA (VSM_WN_TM)	**94.13%**	**99.11%**	**89.15%**	**89.42%**

Table 27 Text categorization average F-1 measure comparison

	Reuters	**TDT2**	**20news**	**VDR**
TD matrix weighted by idf (Baseline)	0.8962	0.8719	0.8587	0.7792
TD matrix weighted by CE_W	0.9093	0.9525	0.8737	0.8349
TD matrix weighted by CE_W + CD matrix (VSM_WN)	0.9144	0.9796	0.8778	0.8503
LTD matrix by PLSA	0.8151	0.8696	0.7881	0.7654
SSTD matrix by semi-supervised PLSA (VSM_TM)	0.8272	0.8820	0.7900	0.8717
Updated TDW matrix weighted by CE_W + CD matrix + LTD matrix	**0.9278**	**0.9902**	**0.8865**	**0.8719**
Updated TD matrix with *tf-CE_W* + CD matrix +semi-supervised PLSA (VSM_WN_TM)	**0.9321**	**0.9915**	**0.8892**	**0.8887**

CHAPTER 6. CONCLUSION AND FUTURE WORK

The major content presented in this dissertation is our research work in the field of typo correction and text categorization, using machine learning, statistical modeling and ontology networks. Specifically, we first propose solutions to automatic typo correction in text documents, in terms of correcting a broad range of typos, from simple typing errors to word boundary errors, unconventional use of acronyms, and multiple versions of abbreviations of the same words. We extract general language knowledge and domain specific knowledge by machine learning algorithms for identifying unconventional acronyms, grouping similar words (correctly spelled and misspelled), and ranking correction candidates. Secondly, we present our research work in generating Vector Space Model (VSM) with ontology networks, as well as latent semantic information generated from statistical topic modeling. Unlike the traditional text representation using only Bag-of-words (BOW) features without considering relationship among words, we utilize semantic and syntactic relationship among words such as synonymy, co-occurrence and context so that the text is represented more accurately.

The results from the performed experiments are highly encouraging. We evaluated our ITDC system through a case study that involves the automatic processing of automotive fault diagnostic text documents. The performance generated from more than 580000 automotive fault diagnostic documents provided by two different automotive manufacturers show that the proposed system outperforms state-of-art spell checking systems. Furthermore, the typo correction process significantly helps improving text categorization performance, by providing more clean and comprehensible text to machines. Moreover, the proposed text representation model is evaluated on three publicly available datasets and a domain-specific dataset. Experiment results show that our approach significantly improves text categorization performance by outperforming baseline approaches such as using only latent features and traditional VSM approaches.

Note here, in the field of typo correction, ITDC system is a big step forward towards fully-automatic spelling correction techniques for processing large size corpora of unstructured text documents. We proposed a general framework for utilize external knowledge from both general and specific domain text resources, using machine learning techniques and statistical analysis. Secondly, we propose a systematic way of building accurate text representation for various text mining applications such as text categorization, text clustering, predictive analysis, information extraction, etc., by capturing not only single word information, but also syntactic and semantic relationship among words. The above ITDC system and VSM_WN_TM model can be combined together for any real-world applications that require text preprocessing and text categorization, and can be easily transplanted and applied to other text corpus, besides those discussed in this dissertation, e.g. text used in social networks such as instant messages and Twitter.

The approaches we proposed for typo correction and text categorization are not perfect without any weaknesses. For the ITDC system, the typo detection, correction and ranking is fully based on pre-built external knowledge and well-prepared training data. Typos in previously unseen documents that are out of the system knowledge scope might not be corrected very well. Also, without sufficient domain-specific knowledge, a lot of non-word typos or domain-specific abbreviations still cannot be easily recognized, so that our ITDC system has to be conservative in correcting typos. This brings a question of how to update the knowledge bases and improve trained system by learning new knowledge incrementally without retraining the whole system, either periodically or in real-time. Also, another interesting direction is to use unsupervised or semi-supervised way in learning new knowledge, especially on specific domains having very large dataset available. It still requires vast amount of work towards the solution of fully automatic and accurate typo correction.

For the VSM_WN_TM model, the proposed text representation approach requires fine-tuned parameters such as number of latent topics, number of maximum iterations, weight assigned to hypernym/hyponym and meronym/holonym relationships, hyper-weight that balance the affection of document content and connectivity, etc. Although they all have optimized range of values based on our experiments, adjustment may still be necessary across different applications in terms of dataset characteristics and user requirements, in order to get the best performance. As a result, efficient and fully automatic approaches could be developed to determine the best set of parameters for specific application. Secondly, information from either ontology networks or statistical topic modeling might also generate additional "noise" features that affect the text categorization accuracy. Instead of concatenate these features together with BOW features, other approaches such as a hierarchical machine learning framework or a combination of multiple classifiers discussed in [119] could be a future direction to perfect our approach. Last but not least, in terms of efficiency, with very large dataset, the system might be slow in finding connections between documents. The next step of work will be redesign our algorithm for parallel computing based on MapReduce platform such as Hadoop [118], so that our approach could be extended to "big text".

REFERENCES

[1] "MIKE2.0, Big Data Definition", web document. URL: http://mike2.openmethodology.org/wiki/Big_Data_Definition

[2] Sholom M. Weiss (Author), Nitin Indurkhya (Author), Tong Zhang (Author), Fred Damerau, Text Mining: Predictive Methods for Analyzing Unstructured Information, Publication Date: November 9, 2004 | ISBN-10: 0387954333, Edition 2005

[3] Qiang Zeng, Xiaoyan Zhang, Weide Zhang, Zuofeng Li and Lei Liu, Extracting Clinical Information from Free-text of Pathology and Operation Notes via Chinese Natural Language Processing, *2010 IEEE International Conference on Bioinformatics and Biomedicine Workshops*, pp 593-597, Hong Kong, 2010.

[4] Yinghao Huang, Naeem Seliya, Yi Lu Murphey and Roy B. Friedenthal, Classifying Independent Medical Examination Reports using SOM networks, *Proceeding of the 6th International conference on Data Mining*, Las Vegas, Nevada, USA, 2010, p58-64

[5] Yinghao Huang, Yi Lu Murphey and Yao Ge, Automotive diagnosis typo correction using domain knowledge and machine learning, *IEEE Symposium Series on Computational Intelligence 2013.*

[6] Karen Kukich, Technique for automatically correcting words in text, ACM Computing Surveys (CSUR), v.24 n.4, p.377-439, Dec. 1992

[7] Christopher Mims, How To Fix Awful Smart-Phone Autocorrection, Web document, URL:
http://www.technologyreview.com/view/424157/how-to-fix-awful-smart-phone-autocorrection/

[8] Dik L. Lee, H.C., Kent Seamons, Document Ranking and the Vector-Space Model, IEEE Software, 1997. 14(2): p. 65-75.

[9] Wongkot Sriurai, IMPROVING TEXT CATEGORIZATION BY USING A TOPIC MODEL, Advanced Computing: An International Journal (ACIJ), Vol.2, No.6, November 2011

[10] R. Feldman and I. Dagan. Kdt - knowledge discovery in texts. In Proc. of the First Int. Conf. on Knowledge Discovery (KDD), pages 112–117, 1995.

[11] U. Nahm and R. Mooney. Text mining with information extraction. In Proceedings of the AAAI 2002 Spring Symposium on Mining Answers from Texts and Knowledge Bases, 2002.

[12] Vishal Gupta, G.Sl Lehal, "A Survey of Text Mining Techniques and Applications", Journal of Emerging Technologies in Web Intelligence, VOL. 1, NO. 1, 60-76, AUGUST 2009

[13] Dik L. Lee, H.C., Kent Seamons, Document Ranking and the Vector-Space Model, IEEE Software, 1997. 14(2): p. 65-75.

[14] Hotho Andreas, Nürnberger Andreas, Paaß Gerhard: A Brief Survey of Text Mining. LDV Forum - GLDV Journal for Computational Linguistics and Language Technology 2005,20(1):19–62.

[15] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. Communications of the ACM, 18(11):613–620, 1975.

[16] S. E. Robertson. The probability ranking principle. Journal of Documentation, 33:294–304, 1977.

[17] Mohammad Ali Elmi and Martha Evens, Spelling Correction Using Context, Proc. 36 th ACL, 17 th COLING, Aug. 10-14 1998, Montreal, Quebec, Canada

[18] Behrang QasemiZadeh, Ali Ilkhani, Adaptive Language Independent Spell Checking using Intelligent Traverse on a Tree, CIS 2006, IEEE

[19] Aminul Islam and Diana Inkpen. 2009. Real-word spelling correction using Google Web IT 3-grams. In EMNLP '09: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 1241–1249, Morristown, USA. Association for Computational Linguistics.

[20] Andrew Carlson and Ian Fette, Memory-Based Context-Sensitive Spelling Correction at Web Scale, Sixth International Conference on Machine Learning and Applications, 2007

[21] Karen Kukich, Technique for automatically correcting words in text, ACM Computing Surveys (CSUR), v.24 n.4, p.377-439, Dec. 1992

[22] Christopher Mims, How To Fix Awful Smart-Phone Autocorrection, Web document, URL: http://www.technologyreview.com/view/424157/how-to-fix-awful-smart-phone-autocorrection/

[23] Qiang Zeng, Xiaoyan Zhang, Weide Zhang, Zuofeng Li and Lei Liu, Extracting Clinical Information from Free-text of Pathology and Operation Notes via Chinese Natural Language Processing, *2010 IEEE International Conference on Bioinformatics and Biomedicine Workshops*, pp 593-597, Hong Kong, 2010.

[24] Yinghao Huang, Naeem Seliya, Yi Lu Murphey and Roy B. Friedenthal, Classifying Independent Medical Examination Reports using SOM networks, *Proceeding of the 6th International conference on Data Mining*, Las Vegas, Nevada, USA, 2010, p58-64

[25] Yinghao Huang, Yi Lu Murphey and Yao Ge, Automotive diagnosis typo correction using domain knowledge and machine learning, *IEEE Symposium Series on Computational Intelligence 2013.*

[26] C. C. Aggarwal and H. Wang. Text mining in social networks. *Social Network Data Analytics*, pages 353–378, 2011.

[27] Johannes Schaback and Fang Li, Multi-Level Feature Extraction for Spelling Correction, IJCAI - Workshop on Analytics for Noisy Unstructured Text Data, pages 79–86, Hyderabad, India.

[28] Phuong H. Nguyen, Thuan D. Ngo, Dung A. Phan, Thu P. T. Dinh, Thang Q. Huynh, Vietnamese spelling detection and correction using Bi-gram, Minimum Edit Distance, SoundEx algorithms with some additional heuristics, 2008 IEEE

[29] Bruno Martins, Mário J. Silva, 'Spelling correction for search engine queries', EsTAL, pp. 372-383, 2004.

[30] Arif Billah Al-Mahmud Abdullah, Ashfaq Rahman, , 'A Generic spell checker engine for south asian languages', accepted for publication and presentation at the IASTED 2003 refereed conference on 'Software Engineering and Applications' ~SEA 2003, Marina del Rey, CA, USA'. Paper No 397-045, November 3-5 2003.

[31] Li Zhuang, TaBao, Xiaoyan Zhu, Chunheng Wang, Satoshi Naoi, 'A Chinese OCR spelling check approach based on statistical language models', International Conference on Systems, Man and Cybernetics, Hague, Netherlands, pp. 4727-4732, IEEE, Oct. 2004.

[32] Victoria J. Hodge and Jim Austin, A Comparison of Standard Spell Checking Algorithms and a Novel Binary Neural Approach, IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 15, NO. 5, 2003

[33] Casey Whitelaw and Ben Hutchinson and Grace Y Chung and Gerard Ellis, Using the Web for Language Independent Spellchecking and Autocorrection, Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 890–899, Singapore, 6-7 August 2009. c 2009 ACL and AFNLP

[34] Stephanie Jacquemont, Francois Jacquenet and Marc Sebban, Correct your text with Google, 2007 IEEE/WIC/ACM International Conference on Web Intelligence

[35] Victoria J. Hodge and Jim Austin, A Comparison of Standard Spell Checking Algorithms and a Novel Binary Neural Approach*, IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING,* vol. 15, No. 5, September 2003.

[36] Monojit Choudhury1, Markose Thomas2, Animesh Mukherjee1, Anupam Basu1, and Niloy Ganguly, How Difficult is it to Develop a Perfect Spell-checker? A Cross-linguistic Analysis through Complex Network Approach, TextGraphs-2: Graph-Based Algorithms for Natural Language Processing, pages 81–88, Rochester, April 2007

[37] E. Brill and R. C. Moore, An Improved Error Model for Noisy Channel Spelling Correction, Proceedings of ACL 2000, Association for Computational Linguistics, Jan 2000.

[38] Kristina Toutanova and Christopher D. Manning, Enriching the Knowledge Sources Used in a Maximum Entropy Part-of-Speech Tagger. *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-2000)*, pp. 63-70, 2000.

[39] Aminul Islam and Diana Inkpen. 2009. Real-word spelling correction using google web it 3-grams. In Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3, EMNLP ’09, pages 1241–1249, Morristown, NJ, USA. Association for Computational Linguistics.

[40] Islam, A. and Inkpen, D. (2009b). Real-Word Spelling Correction using Google Web 1T n-gram dataset. In Proceedings of the 18th ACM Conference on Information and Knowledge Management (CIKM 2009), pages 1689-1692, Hong Kong.

[41] Sebastian van Delden, David Bracewell and Fernando Gomez, Supervised and Unsupervised Automatic Spelling Correction Algorithms, Proceedings of the 2004 IEEE International Conference on Information Reuse and Integration, pp530-535, 2004.

[42] Patrick Ruch, Robert Baud and Antoine Geissbiihler, Toward filling the gap between interactive and fully-automatic spelling correction using the linguistic context, 2001 IEEE International Conference on Systems, Man, and Cybernetics, pp199 – 204, vol.1.

[43] Liping Huang, Yi Lu Murphey: Text Mining with Application to Engineering Diagnostics. IEA/AIE 2006: 1309-1317

[44] Chris Biemann, Ontology learning from text – a survey of methods. LDV-Forum, 20(2):75–93, 2005.

[45] Ding, Y. and S. Foo, Ontology research and development: Part 1 – A review of ontology generation. Journal of Information Science 28(2), 123–136.

[46] A. Hotho, S. Staab, and G. Stumme. Ontologies improve text document clustering. In Proceedings of the International Conference on Data Mining — ICDM-2003. IEEE Press, 2003

[47] B. Shi, et al., Classification of Semantic Documents Based on WordNet, International Conference on E-Learning, E-Business, Enterprise Information Systems, and E-Government, vol. 0, pp. 173-176, 2009.

[48] Litvak, M., Last, M., & Kisilevich, S. (2007). Classification of Web documents using concept extraction from ontologies. Lecture Notes in Computer Science, 4476, pp. 287-292.

[49] WordNet: An Electronic Lexical Database. The MIT Press (1998)

[50] Janik, M., Kochut, K.J.: Wikipedia in Action: Ontological Knowledge in Text Categorization. 2nd International Conference on Semantic Computing (ICSC), Santa Clara, CA, USA (2008)

[51] Buitelaar, P., Cimiano, P., Magnini, B., Ontology learning from text: An overview. Ontology learning from text: Methods, evaluation and applications. Frontiers in Artificial Intelligence and Applications Series 123 (2005).

[52] Cimiano, P. and S. Staab (2004). Learning by googling. SIGKDD Explorations 6(2), 24–34.

[53] Cimiano, P. and S. Staab (2005). Learning concept hierarchies from text with a guided agglomerative clustering algorithm. In Proceedings of the ICML 2005 Workshop on Learning and Extending Lexical Ontologies with Machine Learning Methods (OntoML 05), Bonn, Germany

[54] Burger, S. and Stieger, B., Ontology-based classification of unstructured information, Fifth International Conference on Digital Information Management (ICDIM), pp. 254-259, July 2010.

[55] Jun Fang, Lei Guo and Yue Niu, Documents classification by using ontology reasoning and similarity measure, 2010 Seventh International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), vol. 4, pp. 1535-1539, Aug. 2010.

[56] A SET OF DICTIONARIES FOR AMERICAN AND BRITISH ENGLISH, Patrick W Daly, Max-Planck Institut fuer Aeronomie, URL: http://mirror.utexas.edu/ctan/systems/win32/winedt/dict/english.txt

[57] Pa¸sca, M. (2005). Finding instance names and alternative glosses on the Web: WordNet reloaded. In Proceedings of Computational Linguistics and Intelligent Text Processing: 6th International Conference (CICLing 2005), LNCS 3406, Mexico City, Mexico, 2005, pp. 280–292

[58] Widdows, D. (2003). Unsupervised methods for developing taxonomies by combining syntactic and statistical information. In HLT-NAACL 2003: Main Proceedings, pp. 276–283.

[59] Jun Fang, Lei Guo and Yue Niu, Documents classification by using ontology reasoning and similarity measure, 2010 Seventh International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), vol. 4, pp. 1535-1539, Aug. 2010.

[60] M.-H. Song, S.-Y. Lim, D.-J. Kang, and S.-J. Lee. Automatic classication of web pages based on the concept of domain ontology. Asia-Pacificc Software Engineering Conference, pages 645-651, 2005.

[61] Baeza-Yates, R., Ribeiro-Neto, B, *Modern Information Retrieval*, 1999: Addison-Wesley.

[62] Salton, Gerard. Introduction to Modern Information Retrieval. McGraw-Hill, 1983.

[63] Karen Sparck Jones and Peter Willet, 1997, Readings in Information Retrieval, San Francisco: Morgan Kaufmann, ISBN 1-55860-454-4.

[64] Frakes, W. 1992. Stemming algorithms. In FYakes, W., and Baeza-Yates, R., eds., Information Retrieval: Data Structures and Algorithms. New Jersey: Prentice Hall. 131-160.

[65] Y. Yang, J. O. Pederson. A comparative study on feature selection in text categorization, ACM SIGIR Conference, 1995.

[66] Y. Yang. Noise Reduction in a Statistical Approach to Text Categorization, ACM SIGIR Conference, 1995.

[67] Salton, G., Buckley, C., Term weighting approaches in automatic text retrieval, *Information Processing and Management*, 24(5): 513-523, 1998.

[68] Liping Huang, Yi Lu Murphey: Text Mining with Application to Engineering Diagnostics. *IEA/AIE 2006*: 1309-1317

[69] Dumais, S.T., *Enhancing performance in latent semantic indexing (LSI) retrieval*, in Technical Report Technical Memorandum. 1990. Bellcore.

[70] Raghavan, V. V. and Wong, S. K. M. A critical analysis of vector space model for information retrieval. Journal of the American Society for Information Science, Vol.37 (5), p. 279-87, 1986

[71] Deerwester, S., Dumais, S. T., Furnas, G. W., Thomas, K. L., & Harshman, R. (1990). Indexing by latent semantic analysis. Journal of the American Society for Information Science, 63, 391¨C407.

[72] Landauer, Thomas K., and Dumais, Susan T., Latent Semantic Analysis, Scholarpedia, 3(11):4356, 2008

[73] Bradford, R., An Empirical Study of Required Dimensionality for Large-scale Latent Semantic Indexing Applications, Proceedings of the 17th ACM Conference on Information and Knowledge Management, Napa Valley, California, USA, 2008, pp. 153–162.

[74] Hofmann, T.: Probabilistic latent semantic indexing. In: Proceedings of the 22nd ACM SIGIR, pp. 50–57 (1999)

[75] Dempster, A.P.; Laird, N.M.; Rubin, D.B. (1977). "Maximum Likelihood from Incomplete Data via the EM Algorithm". Journal of the Royal Statistical Society, Series B 39 (1): 1–38. JSTOR 2984875. MR 0501537

[76] Charu C. Aggarwal and ChengXiang Zhai. A survey of text classification algorithms. In Mining Text Data, pages 163-222. Springer, 2012.

[77] B. N. Pandey, N. Dwividi, and B. Pulastya, "Comparison between bayesian and maximum likelihood estimation of the scale parameter in Weibull distribution with known shape under linex loss function," Journal of Scientific Research, vol. 55, pp. 163–172, 2011.

[78] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. J. Mach. Learn. Res., 3:993–1022, 2003.

[79] B S Harish, D S Guru, S Manjunath, Representation and Classification of Text Documents: A Brief Review. IJCA Special Issue on *Recent Trends in Image Processing and Pattern Recognition*, RTIPPR, 2010.

[80] Yinghao Huang, Yi Lu Murphey and Yao Ge, Machine Learning of Engineering Diagnostic Knowledge from Unstructured Verbatim Text Descriptions, , *IEEE Symposium Series on Computational Intelligence,* Singapore, April 15-19, 2013.

[81] Sebastiani, F. 2002. Machine learning in automated text categorization. ACM Computing Surveys. Vol 34, pp. 1 .47.

[82] Hartigan, J.A., Wong, M.A.: A K-Means Clustering Algorithm. Applied Statistics **28,** (1979) 100-108

[83] Jardine, N., Sibson, R.: The construction of hierarchic and non-hierarchic classifications. The Computer Journal 11 (1968)

[84] Dino Isa,, V. P Kallimani Lam Hong lee, "Using Self Organizing Map for Clustering of Text Documents", ", Elsever, Expert System with Applications-2008.

[85] Jingnian Chen a,b,, Houkuan Huang a, Shengfeng Tian a, Youli Qua Feature selection for text classification with Naïve Bayes" Expert Systems with Applications 36, pp. 5432–5435, 2009.

[86] Sang-Bum Kim, Kyoung-Soo Han, Hae-Chang Rim, and Sung Hyon Myaeng, "Some Effective Techniques for Naive Bayes Text Classification", IEEE Transactions On Knowledge And Data Engineering, Vol. 18, No. 11, , Pp- 1457- 1466 ,November 2006.

[87] Y. Yang, C.G. Chute. An example-based mapping method for text categorization and retrieval. ACM Transactions on Information Systems, 12(3), 1994.

[88] A. Y. Ng, M. I. Jordan. On discriminative vs. generative classifiers: a comparison of logistic regression and naive Bayes. NIPS. pp. 841- 848, 2001.

[89] Bo Yu, Zong-ben Xu, Cheng-hua Li ,"Latent semantic analysis for text categorization using neural network", Knowledge-Based Systems 21- pp. 900–904, 2008.

[90] Trappey, A. J. C., Hsu, F.-C., Trappey, C. V., & Lin, C.-I.,"Development of a patent document classification and search platform using a back-propagation network", Expert Systems with Applications, pp. 755–765, 2006.

[91] Cheng Hua Li , Soon Choel Park, "An efficient document classification model using an improved back propagation neural network and singular value decomposition" Expert Systems with Applications 36 ,pp- 3208–3215, 2009.

[92] Bang, S. L., Yang, J. D., & Yang, H. J. , "Hierarchical document categorization with k-NN and concept-based thesauri. Information Processing and Management", pp. 397–406, 2006.

[93] Bao Y. and Ishii N., "Combining Multiple kNN Classifiers for Text Categorization by Reducts", LNCS 2534, , pp. 340- 347, 2002.

[94] Zi-Qiang Wang, Xia Sun, De-Xian Zhang, Xin Li "An Optimal SVM-Based Text Classification Algorithm" Fifth International Conference on Machine Learning and Cybernetics, Dalian,pp. 13-16 , 2006.

[95] Tai-Yue Wang and Huei-Min Chiang "One-Against-One Fuzzy Support Vector Machine Classifier: An Approach to Text Categorization", Expert Systems with Applications, doi: 10.1016/j.eswa.2009.

[96] T. Joachims. A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization. ICML Conference, 1997.

[97] H. Drucker, D. Wu, V. Vapnik. Support Vector Machines for Spam Categorization. IEEE Transactions on Neural Networks, 10(5), pp. 1048–1054, 1999.

[98] D. Mladenic, J. Brank, M. Grobelnik, N. Milic-Frayling. Feature selection using linear classifier weights: interaction with classification models. ACM SIGIR Conference, 2004.

[99] Wongkot Sriurai, IMPROVING TEXT CATEGORIZATION BY USING A TOPIC MODEL, Advanced Computing: An International Journal (ACIJ), Vol.2, No.6, November 2011

[100] Shibin Zhou,Kan Li,Yushu Liu, Text Categorization Based on Topic Model, International Journal of Computational Intelligence Systems, Vol.2, No. 4 (December, 2009), 398-409

[101] Bıro, I. & Szabo, J. (2010) "Large scale link based latent Dirichlet allocation for web document classification"

[102] Bıro, I. & Szabo, J. (2009) "Latent Dirichlet Allocation for Automatic Document Categorization". In Proceedings of the 19th European Conference on Machine Learning and 12th Principles of Knowledge Discovery in Databases.

[103] Yue Lu, Qiaozhu Mei and ChengXiang Zhai, Investigating task performance of probabilistic topicmodels: an empirical study of PLSA and LDA, Inf Retrieval (2011) 14:178–203

[104] Bloehdorn, S., Hotho, A.: Text Classification by Boosting Weak Learners based on Terms and Concepts. 4th IEEE International Conference on Data Mining (ICDM'04)(2004)

[105] Zelikovitz, S., Hirsh, H.: Improving Short Text Classification Using Unlabeled Background Knowledge. Seventeenth International Conference on Machine Learning (ICML), Stanford, CA (2000)

[106] Gradshteyn, I. S. and Ryzhik, I. M. Tables of Integrals, Series, and Products, 6th ed. San Diego, CA: Academic Press, p. 1101-2000.

[107] Vapnyarskii, I.B. (2001), "Lagrange multipliers", in Hazewinkel, Michiel, Encyclopedia of Mathematics, Springer, ISBN 978-1-55608-010-4

[108] D. Cohn and T. Hofmann (2001). The missing link - a probabilistic model of document content and hypertext connectivity. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, Advances in Neural Information Processing Systems 13. MIT Press, Cambridge, MA.

[109] Borgelt, Christian; Kruse, Rudolf, Graphical Models: Methods for Data Analysis and Mining. Chichester, UK: Wiley. ISBN 0-470-84337-3, March 2002.

[110] David Chandler (1987). Introduction to Modern Statistical Mechanics. Oxford. ISBN 0-19-504277-8.

[111] Pei Yang, Wei Gao, Qi Tan, Kam-Fai Wong, A link-bridged topic model for cross-domain document classification, Information Processing and Management 49 (2013) 1181–1193, 2013 Elsevier.

[112] Kristina Toutanova and Christopher D. Manning. 2000. Enriching the Knowledge Sources Used in a Maximum Entropy Part-of-Speech Tagger. In Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-2000), pp. 63-70.

[113] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. (1990). Indexing By Latent Semantic Analysis. Journal of the American Society For Information Science , 41 , 391-407

[114] Deng Cai, Xiaofei He and Jiawei Han, "Document Clustering Using Locality Preserving Indexing", IEEE TKDE 2005.

[115] Reuters-21578, Distribution 1.0. Web document. URL: http://www.daviddlewis.com/resources/testcollections/reuters21578/

[116] Ken Lang, Newsweeder: learning to filter netnews. Proceedings of the Twelfth International Conference on Machine Learning, pp. 331-339, 1995.

[117] C.-C. Chang and C.-J. Lin. LIBSVM : a library for support vector machines. ACM Transactions on Intelligent Systems and Technology, 2:27:1--27:27, 2011.

[118] Chuck Lam, Hadoop in action, Manning Publications, 1 edition. ISBN-10: 1935182196, December 22, 2010.

[119] Bell, D. A. ; Guan, J. W. ; Bi, Y. ; , "On combining classifier mass functions for text categorization, " Knowledge and Data Engineering, IEEE Transactions on , vol. 17, no. 10, pp. 1307- 1319, Oct. 2005, doi: 10. 1109/TKDE. 2005. 167

[120] Dasari, D. B. Rao, V. G. Text categorization and machine learning methods: current state of the art. Global Journal of Computer Science and Tecnology Software & Data Engineering, 12(11):37–46, 2012.

[121] Li T, Zhu S, Ogihara M, Using discriminant analysis for multi-class classification: an experimental investigation. Knowl Inf Syst 10(4):453–472, 2006

[122] R. Bekkerman, R. El-Yaniv, N. Tishby, and Y. Winter, "Distribu¬tional Word Clusters vs Words for Text Categorization, " J. Machine Learning Research, vol. 3, 2003

[123] Vidhya. K.A G.Aghila, “A Survey of Naïve Bayes Machine Learning approach in Text Document Classification”, (IJCSIS) International Journal of Computer Science and Information Security, Vol. 7, 2010.

[124] Methods Ali Danesh Behzad Moshiri “Improve text classification accuracy based on classifier fusion methods”. 10th International Conference on Information Fusion, 2007.