UNIVERSITÉ DE LORRAINE | UFR MATHÉMATIQUES ET INFORMATIQUE

ERASMUS MUNDUS MASTERS PROGRAM IN LANGUAGE AND COMMUNICATION TECHNOLOGY

MASTER THESIS

A graph model for text analysis and text mining

Author:
Thi Ngoc Quynh Do

Supervisor:
Amedeo Napoli (DR CNRS)

June 18, 2012

Abstract

Automated text analysis and text mining methods have received a great deal of attention because of the remarkable increase of digital documents. Typical tasks involved in these two areas include text classification, information extraction, document summarization, text pattern mining etc. Most of them are based on text representation models which are used to represent text content. The traditional text representation method, Vector Space Model, has several noticeable weak points with respect to the ability of capturing text structure and the semantic information of text content. Recently, instead of using Vector Space Model, graph-based models have emerged as alternatives to text representation model. However, it is still difficult to include semantic information into these graph-based models. In this thesis, we propose FrameNet-based Graph Model for Text (FGMT), a new graph model that contains structural and shallow semantic information of text by using FrameNet resource. Moreover, we introduce a Hybrid model based on FGMT which is more adapted to text classification. The experiment results show a significant improvement in classification by using our models versus a typical Vector Space Model.

Keywords: text representation model, graph model, FrameNet, text analysis, text mining

Acknowledgements

It is my great pleasure to acknowledge the people who supported me throughout the writing of this thesis. With my all honour, I give this special thanks to my advisor Dr. Amedeo Napoli, for his warm hospitality during my time at Orpailleur team, LORIA research laboratory in Nancy, France, for his precious lectures on Knowledge Discovery and Knowledge Representation, for many helpful discussions about the topic of this research, and for his invaluable advices in research methodology. Without his help, I wouldn't have completed my study in lovely France. I would like to take this opportunity to thank my lab members for their kindness, helpfulness, and friendliness. I also have to acknowledge the professors at University of Lorraine and Free University of Bozen-Bolzano for their interesting lectures that I took. Furthermore, I would like to thank all professors and staffs of European Masters Program in Language and Communication Technologies for giving me an opportunity to study in an excellent program and for their great supports during my two years of study. Finally, I am grateful to all my dear friends, and especially my beloved family. They are my inspiration, motivation, and incitement for me to walk forward in my life journal.

Contents

List of Tables

List of Figures

Chapter 1

INTRODUCTION

Computer technology has brought a dramatic change to our daily life. Nowadays, by using digital methods, we can store, manage and retrieve information in text documents automatically without looking at printed documents. Automated text analysis and text mining are becoming more and more important in computer applications. Typical tasks involved in these two areas include text classification, information extraction, document summarization, text pattern mining etc. [8]. Most of them are based on text representation models which are used to represent text content so that computer can understand and work with text. Among the current text representation models, Vector Space Model (VSM) is a traditional method used frequently in many tasks because of its simpleness and effectiveness. However, VSM still has several noticeable weak points with respect to the ability of capturing text structure and the semantic information of text content. Recently, instead of using VSM, graph-based models have emerged as alternatives to text representation model. Although several recent researches reported that graph-based models can reach better results than VSM in some classification algorithms [22] [16], these models still need to be improved, especially in the aspect of semantics.

As an approach to deal with the above problems, in this thesis, we propose *FrameNet-based Graph Model for Text (FGMT)* which is a graph text representation model based on frame semantics and FrameNet for text-only documents. This graph model contains structural and shallow semantic information of text content extracted by using semantic role labeling. We also discuss the feasibility of FGMT and its applications in text classification and frequent pattern mining.

A summary of this thesis is as follows: First of all, in *Chapter 1*, we present background in Vector Space Model, text classification, pattern mining, frame semantics, and FrameNet. Secondly, in *Chapter 2*, there is an overview of graph-based techniques used to develop text representation model. Thirdly, in *Chapter 3*, we focus on FrameNet-based Graph Model for Text which is our proposal to include structural and shallow semantic information of text into a graph model by using FrameNet. Next, in *Chapter 4*, an implementation of a tool to build FGMT for a text corpus is presented. Next, in *Chapter 5*, some experiments to evaluate not only the feasibility of FGMT but also its effectiveness in text classification are discussed. Finally, in *Chapter 6*, we summarize the results of this thesis and propose possible future work.

1.1 Vector Space Model - Traditional text representation model

In order to store text documents in computers and allow users to search through their contents, one of the first steps is building a text representation model. There are many ways to represent texts so that they can be "understood" by computers. A typical choice which is used frequently is Vector Space Model.
Salton et al., 1975 [21] presented a VSM as a model in which each text is represented as a vector of *term weights*. A set of *terms* $T = \{t_1, t_2, ..., t_n\}$ that occurred at least once in at least one document, serves as a feature set. Each document d_i is represented by a vector $d_i = (w_{i1}, w_{i2}, ..., w_{in})$ where w_{ij} is the weight of term t_j in document i. Thus, a VSM can be considered as a matrix M_{ij} where terms are in the columns and documents are in the rows. In the matrix, entry m_{ij} is the weight of term t_j in document i. There are various approaches in VSM which are different in definition of term and weighting method [23].
In *term* definition, we can choose different meaningful units of text as terms, such as words or patterns, but words are used widely in traditional approaches [23]. Several linguistic processes like spelling correction and normalization, stopword removal, lemmatization and stemming can be applied in term selection [23]. Especially, when a single word is selected as a term, the set of terms T can be either all or N most weighted words appearing in the collection and this method is called *"Bag of Words"* model.
As to *weighting method*, the *tf-idf*[1] is a popular way, in which *tf* (term frequency) represents the importance of a term in a text and *idf* (inverse document frequency) represents the discrimination of a term for all texts. This measure assigns the highest weight to terms that occur frequently in a specific document but do not occur at all in most other documents. Another typical weighting method is using

[1]http://nlp.stanford.edu/IR-book/html/htmledition/tf-idf-weighting-1.html

Boolean score. In this approach, a document is represented as a vector where dimension values are Boolean with 0 indicating the absence and 1 indicating the presence of the corresponding dictionary term in the document [23].
For example, we consider two texts of "The small dog is beaten by the big cat. Peter likes the dog." and "Mary likes animal." in VSM. After stopword removal, all word lemmas that occur at least once in at least one text and Boolean score are selected as terms and weighting method, respectively. Thus, the list of terms includes "small", "dog", "beat" (lemma of "beaten"), "big", "cat", "Peter", "like" (lemma of "likes"), "Mary", "animal". Our VSM is a matrix M with 9 columns representing twelve terms and 2 rows which describe two texts. If *Term j* occurs in *Text i* then $M_{ij} = 1$ else $M_{ij} = 0$. *Text 1* and *Text 2* are represented by [0 1 1 1 1 1 0 1 1] and [1 0 0 0 0 1 1 0 0], respectively in this simple VSM.

$$\left(\begin{array}{cccccccccc} & animal & beat & big & cat & dog & like & Mary & Peter & small \\ Text\ 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ Text\ 2 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{array}\right)$$

Vector Space Model has some advantages: It is simple and easy to implement. Also, by representing text as a vector of term weights, it can be applied directly in most of machine learning algorithms which are used commonly in text analysis and text mining. However, one of the important disadvantages of this model lies in semantics: It contains not much semantic information.

1.2 Text classification

Text classification is an important technique for organizing text documents into classes. Automatic text classification finds applicability for a number of tasks such as automated indexing of scientific articles, spam filtering, identification of document genre, classification of news articles, etc. For example, in spam filtering, we can use text classification to assign "spam" or "non spam" label to each new message.
In text classification, a text representation model is needed to represent text content. A good text representation model can help to improve classification results. As to text classification algorithms, machine learning is a common approach, which can be divided into supervised classification and unsupervised classification algorithms [23].

1.2.1 Supervised classification algorithms

These algorithms use the training data containing labeled texts, to learn a classifier which classifies new texts. Supervised machine learning techniques like Support Vector Machine, K-Nearest Neighbours, Naive Bayes, Decision Tree, etc. are applied frequently in text classification.

1.2.2 Unsupervised classification (or clustering) algorithms

In unsupervised learning algorithms, we have unlabelled collection of text. The aim is to cluster the texts without additional knowledge or intervention such that documents within a cluster are more similar than documents between clusters. K-Means, Hierarchical clustering, etc. are commonly used as unsupervised learning techniques in text classification.

1.3 Frequent pattern mining

Over the last two decades, the research on mining interesting patterns from databases has received a great deal of attention. Nowadays, there are numerous pattern mining algorithms that can work with variable complexities like set, sequence, tree or graph. These algorithms can be applied in most of the prominent knowledge discovery tasks, like classification, outlier detection, etc. The problem of frequent pattern mining can be defined as follows:
"Given a database D, of a collection of events (an event can be as simple as a set or as complex as a graph) and a user defined support threshold π^{min}, return all patterns (patterns can be set, tree, graph, etc. depending on D) that are frequent with respect to π^{min}" [25].
For text pattern mining, the task is to discover interesting patterns from textual dataset. An approach to mine patterns from text by using graph-based text representation models and frequent subgraph mining can be found in this thesis.

1.4 Brief introduction to Frame Semantics

The historical root of frame semantics in linguistics normally refers to Charles Fillmore's case grammar [6]. In this grammar, a *case frame* was taken to characterize a small abstract scene which identifies (at least) the participants of the scene and thus the arguments of predicates and sentences describing the scene. The language user is supposed to have mental access to such schematized scenes in order to understand a sentence [6].

Frame semantics is an approach to the understanding and description of the meanings of lexical items and grammatical constructions. A main feature of this approach is the slogan of Charles Fillmore: *"Meanings are relativized to scenes"*. That means in order to understand the meanings, we must first have knowledge about the internal structure which is determined with respect to a background frame or a scene. We consider "Commerce buy" frame in "Mary bought a pen" and "Peter bought a pen from Mary" as examples. Here, the concept of frame is applied to the verb "buy" ("bought") and requires obligatorily a BUYER, a GOODS and optionally a SELLER elements: "Mary" is the BUYER in the first sentence, but in the second one, "Mary" is the SELLER. Meanwhile, "a pen" is the GOODS in both sentences and "Peter" is the BUYER in the second sentence. In order to understand the meaning of "bought", we should consider it in the frame with its relative elements.
$[Mary]_{BUYER}\ BOUGHT\ [a\,pen]_{GOODS}.$
$[Peter]_{BUYER}\ BOUGHT\ [a\,pen]_{GOODS}\ [from\,Mary]_{SELLER}.$

1.5 FrameNet - An online lexical resource for English, based on Frame Semantics

FrameNet[2] is the Berkeley FrameNet project [1] which builds an online lexical resource for English based on frames semantics. Its starting point is the observation that words can be grouped into semantic classes, the so-called "frames" (similar to frame in Charles Fillmore's theory). For example, we consider two sentences "Peter likes the dog." and "Mary loves animal.". What is the similarity between these two sentences? The answer is that both of them describe an "*Experiencer*'s emotions with respect to some *Content*" (a scene, or frame). In the first sentence, "likes" is the word evoking the frame, while "Peter" and "the dog" are the Experiencer and Content, respectively. Similarly, "loves" is the word evoking the frame in the second sentence and "Mary", "animal" are Experiencer and Content, respectively. Here, Experiencer and Content are called Frame Element (or Semantic Role).
FrameNet's *ultimate goal* is building a dictionary which links frames to words and the expressions that can introduce them in text.

1.5.1 Methodology

The FrameNet project follows a frame-by-frame basis methodology [1]:
First, *frames* are defined, discovered and described. The core of the process of

[2]https://framenet.icsi.berkeley.edu/fndrupal/

building frame has always been looking at corpus attestations of a group of words that are believed to have some semantic overlap, and dividing these attestations into groups.
Afterward, the small groups are combined into large enough groupings to make reasonable frames. The words used to form frames are called *Lexical Units (LUs), Words Targets (or Targets)*, or *Frame-Evoking elements*. After forming frames, *Frame Elements (FEs)* (or *Semantic Roles (SRs)*) which are participants, props, and other conceptual roles involved in frames are decided.
Finally, examples sentences in data corpus are selected and annotated for each frame.

1.5.2 Components

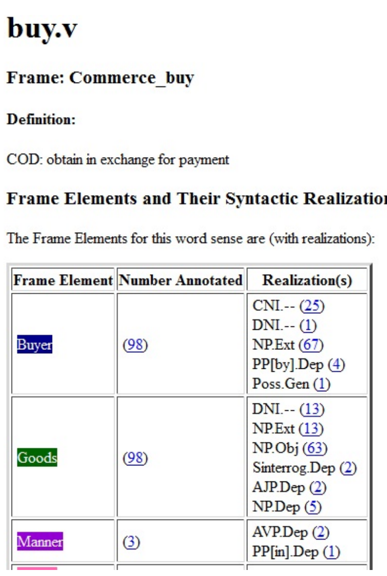
FrameNet contains four main components including Lexical Units, Frames, Frame Ontology and Corpus of examples sentences.

Lexical Unit

A Lexical Unit (LU) in FrameNet is a pairing of a lemma with a meaning (a word sense). Typically, each sense of a polysemous word belongs to a different semantic frame, a script-like conceptual structure that describes a particular type of situation, object, or event along with its participants and props. In addition to the connection to the frame, the FrameNet database also includes a definition of each LU. For example, in Figure 1.1, we show a LU buy.v which belongs to frame Commerce_buy. In this case, LU buy.v is a pairing of a lemma "buy" and the meaning of "obtain in exchange for payment". This LU is linked to the "Commerce_buy" frame. There are several Frame Elements such as "Buyer", "Goods", "Manner" etc. The numbers of annotated examples and the syntactic realization for each Frame Element (FE) are also provided in the LU definition.

Frame

Each frame entry in FrameNet contains a frame definition, a list of Frame Elements (or Semantic Roles), and a set of LUs that can evoke the frame. For instance, in "Commerce_buy" frame which is described in Figure 1.2, Buyer and Goods are two Semantic Roles, and "buy.v" is one of the LU of the frame.
A Frame Element (or Semantic Role) in a frame can be general like Agent, Patient, Theme or more specific like Cogniser, Evaluee, Degree and so on. They normally correspond to a syntactic constituent like noun phrase, verb phrase, adjective phrase, adverb phrase, proposition phrase, clause, etc. [20].



buy.v

Frame: Commerce_buy

Definition:

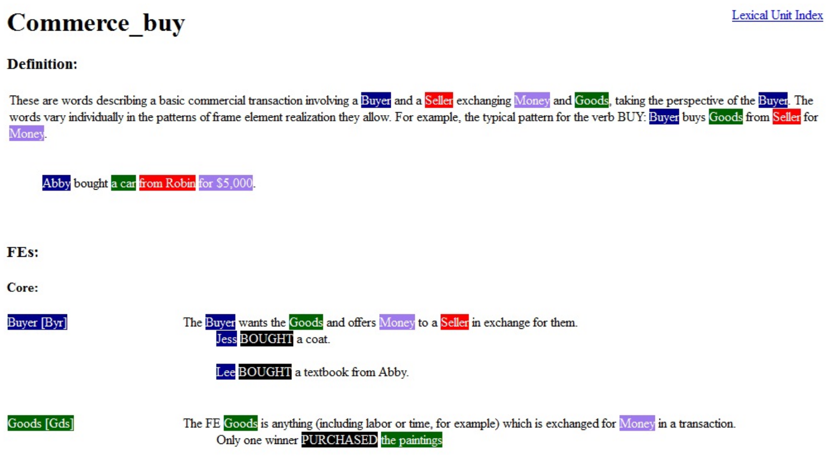
COD: obtain in exchange for payment

Frame Elements and Their Syntactic Realizations

The Frame Elements for this word sense are (with realizations):

Frame Element	Number Annotated	Realization(s)
Buyer	(98)	CNI.-- (25) DNI.-- (1) NP.Ext (67) PP[by].Dep (4) Poss.Gen (1)
Goods	(98)	DNI.-- (13) NP.Ext (13) NP.Obj (63) Sinterrog.Dep (2) AJP.Dep (2) NP.Dep (5)
Manner	(3)	AVP.Dep (2) PP[in].Dep (1)

Figure 1.1: FrameNet lexical entry "buy.v"



Commerce_buy

Lexical Unit Index

Definition:

These are words describing a basic commercial transaction involving a Buyer and a Seller exchanging Money and Goods, taking the perspective of the Buyer. The words vary individually in the patterns of frame element realization they allow. For example, the typical pattern for the verb BUY: Buyer buys Goods from Seller for Money.

Abby bought a car from Robin for $5,000.

FEs:

Core:

Buyer [Byr]	The Buyer wants the Goods and offers Money to a Seller in exchange for them. Jess BOUGHT a coat. Lee BOUGHT a textbook from Abby.
Goods [Gds]	The FE Goods is anything (including labor or time, for example) which is exchanged for Money in a transaction. Only one winner PURCHASED the paintings

Figure 1.2: FrameNet frame "Commerce_buy"

Frame Ontology

Frames in FrameNet are connected to each other via frame-to-frame relations and these relations can be read from a frame ontology. Several types of relations are defined, of which the most important are “Inheritance”, “Using”, “Subframe” and “Perspective on” [20].
Inheritance (an IS-A relation) - The child frame is a subtype of the parent frame, and each FE in the parent is bound to a corresponding FE in the child. For instance, the “Revenge” frame inherits from the “Rewards and punishments” frame.
Using - The child frame presupposes the parent frame as background, such as the Speed frame “uses” (or presupposes) the Motion frame, however, not all parent FEs need to be bound to child FEs.
Subframe - The child frame is a sub event (or situation, object) of a complex event (or situation, object) represented by the parent. For example, the “Criminal process” frame has subframes of “Arrest”, “Arraignment”, “Trial”, and “Sentencing”.
Perspective on - The child frame provides a particular perspective on a parent frame. A pair of examples consists of the “Hiring” and “Get a job” frames, which perspectivize the “Employment start” frame from the “Employers scenario” and the “Employees scenario” point of view, respectively.

1.5.3 Public data

The corpus of examples annotated in FrameNet includes sentences from British National Corpus[3] (FrameNet I) and LDC North American Newswire corpora[4] (FrameNet II) which are annotated by FrameNet group. FrameNet’s public data contains three main layers (Frame Element layer, Grammatical function layer and Phrase Type) for all of the annotated sentences, list of frames and FE descriptions, Frame-to-frame relations and Lexical entries summarizing the valence patterns for each annotated LU.
The current release of FrameNet, Release 1.5, has the size of 11829 lexical units, 1019 frames, 173018 annotated sentences. The major change of the newest release is in data format and it causes troubles to long-time users of the FrameNet data. For more detail, see the release document of FrameNet 1.5.

[3]http://www.natcorp.ox.ac.uk/
[4]http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC95T21

1.5.4 A comparison with PropBank

Besides FrameNet, PropBank (Palmer, Gildea, and Kingsbury, 2005, [19]) is also one of the major projects that produced text with semantic role (which captures the relationship between a predicate and syntactic constituents) annotation. It is a 300.000-word corpus of Penn Treebank's data[5] tagged with predicate-argument relations. Propbank has much larger annotation coverage than FrameNet (FrameNet has 11829 lexical units). The goals of PropBank are to obtain a complete semantic role annotation of the Penn Treebank and to provide consistent argument labels across different syntactic realizations of the same verb. In Propbank annotation, verb and its semantic roles can be considered as predicate and arguments, respectively. A *key difference* between FrameNet and PropBank is in the set of semantic roles. Propbank uses more general roles and has a lower number of roles. It has five numbered main roles which are A0 (Agent), A1 (Patient), A2 (indirect object), A3 (start point), A4 (end point) and several other arguments like ArgM-TMP (Temporal marker), ArgM-LOC (location), ArgM-DIR (direction), ArgM-MNR (Manner). For example, in the same sentence which is annotated in FrameNet as $[Mary]_{BUYER}$ $BOUGHT$ $[a\ pen]_{GOODS}$., the annotation in Propbank is $[Mary]_{A0}$ $BOUGHT$ $[a\,pen]_{A1}$ where BOUGHT is the predicate, "Mary" and "a coat" are two arguments named by "A0" and "A1". Because of this key difference, we chose FrameNet for our work as a linguistic resource containing a rich set of semantic roles.

1.6 Synthesis

VSM is a simple method that has shown its good effectiveness in many applications like information retrieval, text classification. However, VSM still has several disadvantages which lie especially in semantic aspect. In almost VSM methods, terms are weighted by occurrence, frequency or some other statistic information, but not semantics. Meanwhile, structural and semantic information is quite critical in getting accurate text classification. Thus, we believe that it is necessary to develop a new text representation model that can capture better these kinds of information. In this thesis, we study graph-based models as alternative choices for text representation. Especially, by using FrameNet which is a semantic linguistic resource, we propose a *FrameNet-based Graph Model for Text* which is a graph-based text representation model containing structural and shallow semantic information of text. We also discuss the feasibility and applications of this model in text classification, and pattern mining tasks.

[5]http://www.cis.upenn.edu/ treebank/

Chapter 2

GRAPH-BASED TEXT REPRESENTATION MODELS

In this chapter, a state of the arts of graph-based text representation models is presented. We also summarized several different approaches to apply graph-based models in text classification.

2.1 Introduction

In chapter 1, we have already seen that the traditional text document representation technique, Vector Space Model, does not take into account the semantic and structural information of text. However, these information types may play a crucial role in text mining tasks, so it raises the problem of how to define an alternative to text representation model. Graph-based text representation techniques were developed to overcome this problem.
In graph text representation models, a text is represented as a graph containing a set of vertices (nodes) and a set of edges representing relationships between nodes. Although the use of graphs for representing text has a very long history in Natural Language Processing (NLP), it has focused on language understanding techniques such as part of speech tagging, rather than text mining tasks like text classication [10]. Recently, some work considering document classification as the objective of graph-based text representation techniques has been done. In this chapter, we give a brief introduction about these graph-based models and their application in text classification.

2.1.1 Some basic definitions on graphs

A *labeled graph* G is a 4-tuple: $G = (V, E, \alpha, \beta)$, where V is a set of vertices, and $E \subseteq V \times V$ is a set of edges that connect the vertices, $\alpha : V \to L_v$, $\beta : V \times V \to L_e$ are vertices labeling functions, and edges labeling functions, respectively (with L_v and L_e are the sets of labels that can appear on the vertices and edges). We may refer to G as G = (V , E) by omitting the labeling functions.
A graph $G_1 = (V_1, E_1, \alpha_1, \beta_1)$ is a *subgraph* of a graph $G_2 = (V_2, E_2, \alpha_2, \beta_2)$, denoted $G_1 \subseteq G_2$, if $V_1 \subseteq V_2$, $E_1 \subseteq E_2 \cap (V_1 \times V_1)$, $\alpha_1(x) = \alpha_2(x)\ \forall x \in V_1$, and $\beta_1(x, y) = \beta_2(x, y)\ \forall (x, y) \in E_1$. Conversely, graph G_2 is called a *supergraph* of G_1.
There are several different types of graph. An *undirected graph* is one graph in which edges have no orientation. Therefore, the edge (a, b) is identical to the edge (b, a). In contrast, a graph that has directed edges is called a *directed graph* or sometimes just a *digraph.* Meanwhile, the term *multigraph* refers to a graph in which multiple edges between nodes are either permitted or required. Another common type is *weighted graph* which is a graph in which each edge has an associated numerical value, called a weight. Usually, the edge weights are non-negative integers. Weighted graphs may be either directed or undirected.

2.1.2 Frequent subgraph mining

As a field of frequent pattern mining (see Section 1.3), Frequent Subgraph Mining (FSM) deals with databases of graphs. Because of the ease with which data can be represented as graph formats, there has been much interest in the mining of graph data. The objective of FSM is to extract all the frequent graph subgraphs in a given dataset, whose occurrence counts are above a specific threshold. For a survey, see for instances the papers by Jiang et al., 2004 [11], Lakshmi et al., 2012 [12].
The problem of FSM can be defined as follows:
"Given a graph dataset $D = \{G_0, G_1, ..., G_n\}$, support$(g)$ denotes the number of graphs (in D) in which g is a subgraph. The problem of frequent subgraph mining is to find any subgraph g such that support$(g) \geq minSup$ where minSup is a minimum support threshold" [26].
Various approaches have been applied to deal with this problem. An Apriori-based algorithm used to discover all frequent (both connected and disconnected) substructures was proposed by Inokuchi et al., 2000 [9]. Kuramochi and Karypis, 2001 [13] developed FSG, a method using adjacent representation of graph and an edge-growing strategy to find all frequent connected subgraphs. In another work, Xifeng Yan and Jiawei Han, 2002 [26] proposed gSpan which is the first algorithm

that explores depth first search in frequent subgraph mining.
In this thesis, we consider to apply FSM to our graph models for pattern mining and text classification tasks.

2.1.3 Graph as text representation model

There is a variety of information types, which can be used to construct graph describing text, such as morphological, syntactic, and semantic features. Some basic types including word forms, lemma, stem, part of speech etc., have applied commonly in graph models. Meanwhile, word orders, word locations or syntax structure are considered as structural information. In term of semantics, several simple semantic information types like synonym, hypernym are taken into account. However, it is quite difficult to capture a deeper semantic meaning of a text.
These above information types may be combined in different ways to create different types of graph representations:
Schenker et al., 2005 [22] proposed *Graph Models for Web Documents (GMWDs)* including several methods for representing web document content (or text documents in general) as graphs. They also proposed several distance and similarity measures between graphs for classication and reported a signicant improvement in classication accuracy achieved with graph versus bag of words representation. However, running algorithms on these graphs were found to be much slower than on vectors. Another problem of graph representation is that documents represented by graphs cannot be classified with most model-based classifiers [16]. In order to overcome these issues, some authors have suggested the use of frequent subgraph mining to build a hybrid model [16][15][14][10]. In this approach, frequent subgraph mining is applied to find a list of subgraphs among graphs representing a set of text documents. Afterward, these subgraphs can be used as terms as in Vector Space Model (see the notion of terms in section 1.1), and then a document is represented as a vector of term weights. Models in this approach can be called *Hybrid* models. Regarding semantic aspect, another type of graph model is given by *Conceptual Graphs (CGs)* [24] which can describe semantic relations between words. However, it is quite difficult to transform natural language text to conceptual graph structures [18].

2.2 Graph Models for Web Documents

Schenker et al., 2005 [22] proposed Graph Models for Web Documents (or text documents in general) including 6 methods of creating graphs from web documents: *Standard, Simple, n-Distance, n-Simple Distance, Absolute Frequency and Relative Frequency.* All of those graph representations are based on the adjacency of terms in an HTML document.

2.2.1 Standard representation

Under the standard representation, the first task is to identify terms, which can be stems, lemmas etc., by using stemming algorithm or other language-specific normalization techniques, then each unique term appearing in the document becomes a node in the graph representing that document. Each node is labeled with the term it represents. The node labels in a document graph are unique because a single node is created for each term, even if a term appears more than once in the text. Second, if a word A immediately precedes a word B somewhere in a "section" (*text content, title, or link etc.*) S of the document, then there is a directed edge from the node corresponding to term A to the node corresponding to term B with an edge label B. An edge is not created between two words if they are separated by certain punctuation marks. With this representation, the graph can capture structural information of text (location, relative location of words). There are three sections defined for standard representation including title, link and text. *Title* contains the text related to the documents title and any provided keywords (metadata). *Link* is the anchor text that appears in hyperlinks on the document. *Text* comprises any of the visible text in the document (this includes hyperlinked text, but not the text in the documents title and keywords). Graph representations are language independent meaning that they can be applied to a normalized text in any language.
An example of a standard graph representation for a short English Web document having the title "SPORT NEWS", a link whose text reads "MORE NEWS", and text containing "ENGLAND FOOTBALL NEWS", is shown in Figure 2.1, where TL denotes the title section, L indicates a hyperlink, and TX stands for the visible text. There are five words occurred in the document: "SPORT", "NEWS", "MORE","ENGLAND", "FOOTBALL", which correspond to five nodes in the graph. Four edges in graph show the relations between words in the documents: for instance, there is an edge from "SPORT" to "NEWS" labeled by "TI" meaning that "SPORT" immediately precedes "NEWS" in the title section.

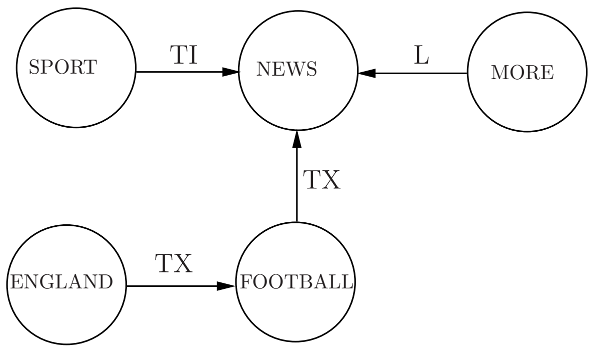


Figure 2.1: Example of a standard graph representation of a document

2.2.2 Simple representation

The second type of Schenker's graph representation is called the simple representation which is basically the same as the standard one, except that no title or meta-data is examined and the edges in the graph are not labeled.

2.2.3 n-Distance representation

The third representation type is n-distance representation. Instead of considering only terms immediately following a given term in a web document, we look up to n terms ahead and connect the succeeding terms with an edge that is labeled with the distance between them (unless the words are separated by certain punctuation marks). For example, in the graph of the text "ENGLAND FOOTBALL NEWS", there are an edge from "ENGLAND" to "FOOTBALL" labeled with 1, an edge from "ENGLAND" to "NEWS" labeled with 2 and an edge from "FOOTBALL" to "NEWS" labeled with 1. The graph for this example is shown in Figure 2.2.

2.2.4 n-Simple distance representation

The fourth graph representation, n-simple distance is similar to n-distance. This is identical to n-distance, but the edges are not labeled meaning that we only know that the distance between two connected terms is not more than n.

2.2.5 Absolute frequency representation

The fifth graph representation is called the absolute frequency representation. This is similar to the simple representation but each node and edge is labeled with an additional frequency measure. For nodes, this indicates how many times

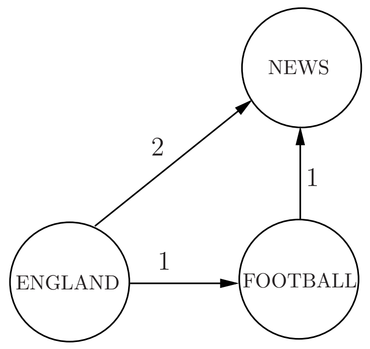


Figure 2.2: Example of a n-distance graph representation of a document

the associated term appeared in the web document. For edges, this indicates the number of times the two connected terms appeared adjacent to each other in the specified order.

2.2.6 Relative frequency representation

The final graph representation is the relative frequency representation, which is the same as the absolute frequency representation but with normalized frequency values associated with the nodes and edges. The absolute frequency representation uses the total number of term occurrences (on the nodes) and co-occurrences (edges).

2.2.7 Application in text classification

After representing text documents in graph models, a text corpus becomes a graph corpus. Schenker et al., 2005 [22] also discussed several approaches to calculate distance and similarity measures between graphs for classification such as graph edit distance, distance based on maximum common subgraph/minimum common supergraph, state space search approach, probabilistic approach, etc. By using similarity measures between graphs, we can apply several machine learning methods (which can work by using similarity measures between objects) on the graph corpus data. The authors showed that the graph representation scheme under the k-Means algorithm compares favorably with the vector approach in terms of clustering performance. Also, a signicant improvement in k-NN classication accuracy achieved with graph versus bag-of-words representation is reported in [22].

2.2.8 Some issues

Although these graph models have the capability of capturing some kinds of structural information (position, relative location of words) in texts, they do not take into account the syntactic structure and semantic relations between words.
With regard to text classification, these models can reach a higher accuracy in some methods like k-NN, k-Means in comparison to VSM. However, there are several issues when running machine learning algorithms on graph models: First of all, it is more time-consuming than running on vector models. Moreover, documents represented by graphs cannot be classified with most model-based classifiers like Decision Tree, Naive Bayes [16].

2.3 Hybrid models

In order to overcome the issues of graph models, hybrid models which uses frequent subgraph mining, have been proposed in several works [16] [15] [14] [10]. The main idea of this model is that after representing all documents in Graph-based models, we use the retrieved graphs to represent documents a way that is similar to Vector Space Model: We consider hybrid representation model as a matrix, in which, terms (see section 1.1) are columns, documents are rows, matrix entries are weights of terms in documents. Given a corpus of n text documents $C = \{d_0, d_1, ..., d_n\}$ as input, the steps to construct a hybrid model are as follows:
Firstly, we represent the text corpus in a graph model like Graph Models for Web Documents (see Section 2.2). After this step, we retrieve a graph corpus $G = \{G_1, G_2, ..., G_n\}$ in which each document d_i is represented by a graph G_i.
Secondly, in order to define *terms* (columns), frequent subgraph mining (see Section 2.1.2) is applied to the graph corpus retrieved in the first step to select subgraphs that occur frequently in the graph corpus. A list of m subgraphs $T = \{g_1, g_2, ..., g_m\}$ (g_j is a frequent subgraph of the graph corpus G) can be used as terms.
Finally, a weighting method is defined to evaluate the weight of a term in a document. A simple weighting method can be used is Boolean score [16]. With this weighting method, each text d_i is represented by a vector $v_i = \{vg_{i1}, vg_{i2}, ..., vg_{im}\}$ where $vg_{ij} = 1$ if g_j is a subgraph of G_i and $vg_{ij} = 0$ if g_j is not a subgraph of G_i.
Regarding *frequent subgraph mining algorithms*, to improve the result of classification, Markov et al., 2008 [16] proposed several methods which are based on FSG algorithm [13], but the extraction steps are done with respect to document classification. In another approach, instead of using un-weighted graphs, Jiang et al., 2010 [10] presented weighted frequent subgraphs, WgSpan, and how it is integrated into the classication process.

In conclusion, Hybrid models have two main benefits [16]: (1) First of all, they keep the important structural information by extracting relevant subgraphs from a graph that represents the document. (2) Secondly, they can be applied in most model-based classification algorithms for inducing a classication model because, eventually, a document is represented by a simple vector. However, the semantic information captured in a hybrid model depends on the graph representation used to construct the hybrid model. In the next section, we introduce another type of graph model that has better ability to capture semantic meaning.

2.4 Conceptual Graphs

There are some emerging approaches of using more complete representations of texts than just words and simple relations between words. One of the common methods to capture the semantic relations between words is given by Conceptual Graphs (CGs), the model introduced by Sowa, 1984 [24]. In CGs, there are two types of nodes which are Concepts and Relations. Among them, a Relation node indicates the semantic role of the incident concepts. For instance, the sentence *"Mary is wearing jeans"* can be represented as a conceptual graph as in Figure 2.3. The rectangles and circles in the graph are Concepts and Relations, respectively. An arc pointing toward a circle marks the first argument of the relation, and an arc pointing away from a circle marks the last argument. Here, *Wear* is generic concept while *Mary* and *Jean* are individual concepts of *PERSON* and *CLOTHING*. The relations from *Wear* to *Mary* and from *Wear* to *Jeans* are named *Agent*, *Object*, respectively, because *Mary* and *Jeans* play *Agent* and *Object* semantic roles in the current context.

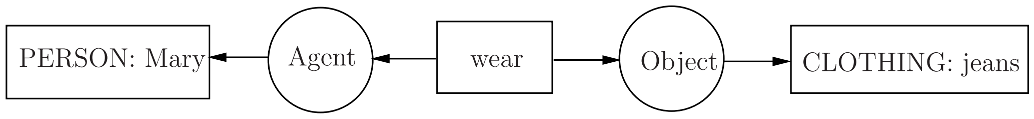


Figure 2.3: An example of Conceptual Graphs

Conceptual Graphs contains rich semantic information, so they can be used in knowledge representation. A semantic meaning of a sentence can be obtained by translating CGs to predicate calculus. The official standard for conceptual graph syntax and semantics is the ISO/IEC 24707 standard for Common Logic, which defines the semantics in terms of an abstract syntax and model-theoretic semantics[1]. However, it is not easy to transform natural language text to CGs

[1]http://www.jfsowa.com/cg/cgdpansw.htm

structures [18]. There are several works in CGs construction which can be divided into manual development, deterministic approaches and statistical approaches [18]. As an example of deterministic approach, Hensman et al., 2005 [7] described the semi-automatic construction of conceptual graph representations of texts using a combination of existing linguistic resources, such as VerbNet and WordNet. The main idea of this method is that the authors used VerbNet and WordNet to identify semantic roles. First, all documents were converted into XML format. Then, they used a syntactic parser to parse all the sentences and identify roles using VerbNet. For each clause in the sentence, the main verb was identified and a sentence pattern is built using parse tree. For each verb in the sentence, they extracted all the possible semantic frames from VerbNet. Finally, the conceptual graph for each sentence was built according to standard rules of CGs. In another work, Sonia et al., 2010 [18] proposed the use of grammar based on the dependency formalism and the standard defined for Conceptual Graphs. The authors used noun pre-modifiers and noun post-modifiers, as well as verb frames, extracted from VerbNet, as a source of definition of semantic roles to built the a dependency grammar, which included verb classification, their syntactic description and frame descriptions. This grammar was designed for the obtained trees to resemble CGs. In brief, by using CGs, a rich semantic information of a text can be captured to a graph, but the fact remains that building this type of graph is not a simple task.

Chapter 3

A FRAMENET-BASED GRAPH MODEL FOR TEXT

In this chapter, we present FrameNet-based Graph Model for Text (FGMT), a model to represent text document as a graph which contains shallow semantic information.

3.1 Motivation for a new graph-based text model

Graph Models for Web Documents (see Section 2.2) have the ability to capture more structural information of text than Vector Space Model and have shown significant improvement in classification (k-NN, k-Means) accuracy in comparison to VSM. However, these models can not be applied directly in most model-based classifiers like Decision Tree, Naive Bayes. Also, using these models in text classification are time-consuming processes because of complexity issues related to the computation of similarity measure between graphs. Furthermore, one of the most important weakness of these models is that they cannot capture much semantic information. In another approach, *Hybrid models* can be considered as a solution to reduce the problems of Graph Models for Web Documents, but it does not resolve semantic issues (see Section 2.2, 2.3). Meanwhile, although Conceptual Graphs contain rich semantic information, it is not easy to transform natural language to this type of graph (see Sections 2.4). Taking into account all of these problems, in this chapter, we propose a method able to represent a text as a graph with shallow semantic information and which is simpler than Conceptual Graphs.

3.2 Method overview

The objective of our method is to represent a text as a graph which contains semantic information of the text. By considering a text as a collection of frames in FrameNet format, we first construct a graph for each frame in the text, then combine all of the obtained graphs into a single graph representing the whole text. Given *a text as input*, in Figure 3.1, we show an overview of our method which contains three main steps: (1) *Shallow Semantic Analysis*: This is the first main step of our method which has the goal of annotating text with semantic frames based on FrameNet. (2) *Graph Construction*: The main function of this step is to build graphs representing the frames detected from text in the first step. (3) *Graph Completion*: To build graph representing the text, we combine frames graphs constructed in the previous step into a single one. The *output of our method is a single graph* describing the given text.
In Figure 3.1, given the text "The small dog is beaten by the big cat. Peter likes the dog.", two frames "Beat_opponent" and "Experiencer_focus" along with their elements are identified in Shallow Semantic Analysis step. Afterward, Graph Construction step builds two graphs representing two frames. Finally, these two graphs are combined into a single graph describing the whole text in Graph Completion step.

3.3 Shallow Semantic Analysis

This step aims to label new, unrestricted text with semantic frames and role information based on FrameNet. For example, given the text "The small dog is beaten by the big cat. Peter likes the dog." as in Figure 3.1, "Beat_opponent" and "Experiencer_focus" frames are detected. Frame "Beat_opponent" has "beat", "the big cat" and "the small dog" as target, "Winner" role and "Loser" role, respectively. Meanwhile, in "Experiencer_focus", "like" is the target, "Experiencer" and "Content" are two roles assigned to "Peter" and "the dog", respectively. In NLP, this task can be referred to *Shallow Semantic Parsing* or *Semantic Role Labeling* which has become a leading task in computational linguistics recently [3],[17]. In order to solve this task, we use Shalmaneser[1] which is a free toolchain for shallow semantic parsing and adapt it to our work.

[1]http://www.coli.uni-saarland.de/projects/salsa/shal/

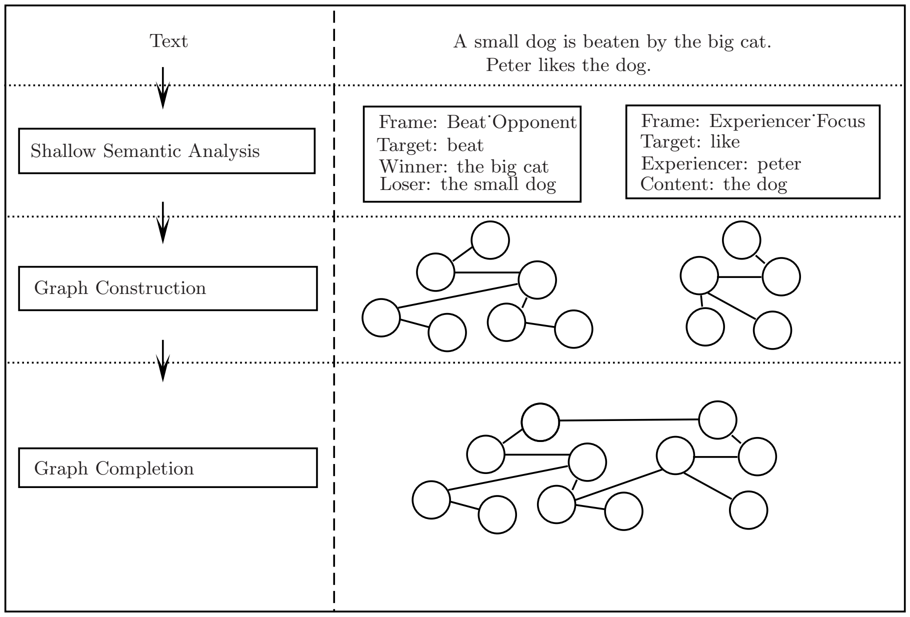


Figure 3.1: Method overview of FGMT

3.3.1 Shalmaneser

Although there has been a lot of fundamental research on the task, starting from Gildea and Jurafsky (2002) [5], up to the shared tasks at CoNLL[2] and SENSEVAL[3] 3, only few Shallow Semantic Parsing tools are available freely for FrameNet[4] [3] such as Shalmaneser, LTH System for Frame-Semantic Structure Extraction[5] and Semafor[6]. Among them, Shalmaneser [3] is a tool that was designed for easy integration with other NLP tools.
Shalmaneser is a loosely coupled toolchain which has modular architecture, so it enables the integration of additional processing modules. Furthermore, the processing components have been kept encapsulated to be easily adaptable to new features, parsers, languages, or classication algorithms.
Shalmaneser has *three components* of Preprocessing, Frame Disambiguator and Role Assignment System. These modules use SALSA/TIGER XML [4] which is

[2]http://www.cnts.ua.ac.be/conll2004/,http://www.lsi.upc.edu/ srlconll/
[3]http://www.senseval.org/senseval3
[4]https://framenet.icsi.berkeley.edu/fndrupal/
[5]http://nlp.cs.lth.se/software/
[6]http://www.ark.cs.cmu.edu/SEMAFOR/

a powerful and versatile XML format for representing semantic roles of Saarland univeristy as a interchange format.
Pre-processing: Parsers, lemmatizers and part of speech taggers are used to pre-process data and export output in SALSA/TIGER XML format.
Frame Disambiguator: It is a module for identifying target (the lexical unit evoking a frame) and assigning a correct frame to the target (each target may be linked to more than one frames because of word sense ambiguation problem). In fact, the first task, identifying target candidate is a simple task because FrameNet provides a list of Lexical Units which are targets of frames. The major problem here is assigning a correct frame to the target, and it is resolved by using *supervised machine learning approach* (Naive Bayes). This module uses a rich set of features for machine learning algorithm such as a bag of words context, with a window size of one or more sentences, bigrams and trigrams centered on the target word, grammatical functions of the target word etc.
Role assignment system: It assigns semantic roles to the linguistic context of a target, based on the semantic frame assigned to the target. The task can be performed in one step, or it can be split into argument recognition (argrec) and argument labeling (arglab). Argrec step distinguishes only between roles and non-roles, while Arglab performs a more detailed classication on the instances recognized as roles in the first step. In this module, *supervised machine learning approach* (Maximum Entropy method) is also applied. Currently, this module includes 30 features for machine learning algorithm.
When testing on FrameNet 1.2 data (90% training, 10% testing), Shalmaneser can reach 93.2% accuracy for frame disambiguation and 85.5% Precision, 66.9% Recall for role assignment.
At this current version (release 1.1), Shalmaneser supports English pre-trained classifiers for FrameNet 1.3.

3.3.2 Our Shallow Semantic Analysis

In our Shallow Semantic Analysis step, Shalmaneser is used as a free tool for shallow semantic parsing. In order to adapt new parsers, lemmatizers and part of speech taggers to Shalmaneser, we designed our *own preprocessing module* which includes part of speech tagging, named entity recognition and syntax tagging. Each sentence is represented as a list of tokens which are annotated by part of speech tags, syntax categories, named entity tags, and several morphology features like lemma and word form. Currently, we use Standford CoreNLP[7] and GATE[8]

[7]http://nlp.stanford.edu/software/corenlp.shtml
[8]http://gate.ac.uk/

as tools for preprocessing tasks. The output of our preprocessing module is in Tiger/Salsa XML format so that it can work easily with Frame Disambiguator and Role assignment system of Shalmaneser.
Another problem when working with Shalmaneser is that it supports only data format of FrameNet 1.3 which is different to the data format of the newest version, FrameNet 1.5 (see section 1.5). In order to solve this problem, we created new pre-trained classifiers for FrameNet 1.5: all examples sentences in Lexical Units data of FrameNet 1.5 are analyzed by our own preprocessing module, then trained by Frame Disambiguator and Role Assignment System of Shalmaneser. By using the new pre-trained classifiers, Shalmaneser now has the ability to work with FrameNet 1.5.

3.4 Graphs construction

After Semantic Analyser step, we retrieve a set of frames (in FrameNet format) together with their targets (the lexical units evoking frames, see Section 1.5) and semantic roles. The aim of this Graphs construction step is developing a method to construct a graph representing a FrameNet frame.
A FrameNet frame consists of one target (the lexical unit evoking frame) and several semantic roles assigned to syntactic constituents. Therefore, in order to build a graph for frame, we first build graphs for semantic roles, and then use the obtained graphs to construct a graph for the whole frame. For example, the text "the small dog is beaten by the big cat." has a frame "Beat_Opponent" with "beat" as target, "the small dog" and "the big cat" as two semantic roles named by "Loser" and "Winner", respectively. To construct a graph for frame "Beat_Opponent", we first build two graphs representing "the small dog" and "the big cat", then combine these two graphs to build graph for the whole frame.

3.4.1 Graph for semantic role

Regarding the fact that each semantic role corresponds to a syntactic constituent, building a graph for a semantic role can be consider as constructing a graph for a syntactic constituent. The main idea is that, based on the syntax pattern of the given syntactic constituent, we determine which elements are important so that they should be chosen as nodes in the graph and what relations between them are. It is similar to the idea of using the syntax pattern to build conceptual graph (see Sowa, 1984 [24] and Hensman et al., 2005 [7]). However, in this work, we simplify the task to building a graph which is simpler than conceptual graph.
First of all, for each of syntactic constituent corresponding to the semantic role,

we extract its syntax pattern (can be obtained from syntactic parsing result) that forms the constituent. For instance, the sentence "Mary and her husband like animal" has "Experiencer_focus" frame in which "like" is a target, "Mary and her husband" and "animal" are "Experiencer" and "Object" semantic roles, respectively. The syntactic parsing result of this sentence can be visualized by a syntax tree as in Figure 3.2. "Mary and her husband" corresponds to the syntax constituent NP_1 and we write the syntax pattern forming this constituent as $NP > NP\ CC\ NP$. The left side of the pattern, NP, is the syntactic category of "Mary and her husband". Meanwhile, the right side of the pattern consists of three components NP, CC and NP corresponding to the syntactic categories of "Mary", "and", and "her husband", respectively. The syntax pattern means that "Mary and her husband", a NP, is formed by three components including a NP, a CC and another NP. In a special case, NP_3 is a syntactic constituent corresponding to "Mary", but it is formed directly from only the word "Mary". Here, we write its syntactic pattern as "$NP >$ " that has no component in the right side.

In the next step, after extracting the syntactic pattern of semantic role, we build

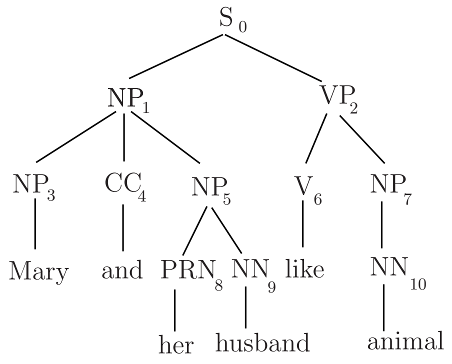


Figure 3.2: Syntax tree of "Mary and her husband like animal"

a graph for the semantic role by using a set of rules. These rules are similar to rules in Conceptual Graph theory but we do not use the notions of Concept and Relation. Each rule corresponds to a syntactic pattern and shows how to build graph from the pattern. We know that the syntactic category of a semantic role in FrameNet can be Noun Phrase (normally), Prepositional Phrase, Verb Phrase, Adjective Phrase, Adverb Phrase, or Clause etc [20]. Therefore, the syntax patterns we have to deal with are the ones forming the above syntactic categories. In order to build the rule set, we collect syntactic patterns from data corpus, then

solve each pattern individually. For each pattern, we determine important components that can be represented as nodes in the graph and the relations between them. For example, given the semantic role "the big cat", the syntax pattern is $NP > DET\ JJ\ NN$ where NP is a Noun Phrase (the big cat), DET is a Determiner (the), JJ is an Adjective (big), NN is a Noun (cat). Taken into account the fact that in Conceptual Graphs, Noun and Adjective are normally represented as Concepts and then are linked by Relations [24], the rule for this pattern can be defined as: the graph for the syntactic constituent formed from the pattern $NP > DET\ JJ\ NN$ consists of two nodes, one node represents the JJ and one node represents the NN. Also, there is an edge from the node represented NN to the node represented JJ. Thus, the graph for "the big cat" has two nodes labeled with "big", and "cat", and one edge from "cat" to "big". The rules used in this step are in the following format:
Pattern: $cat_1 > cat_2\ cat_3\ ...\ cat_n$
ID: $id_1 > id_2\ id_3\ ...\ id_n$
Heads of Graph: $id_{h1}, id_{h2}, ..., id_{hk}$
Nodes of Graph: $id_{i1}, id_{i2}, ..., id_{iq}$
Edges of Graph:
$id_{e11} - id_{e12} - label_1$
$id_{e21} - id_{e22} - label_2$
...
$id_{et1} - id_{et2} - label_t$
This rule is used to build graph for the pattern $cat_1 > cat_2\ cat_3\ ...\ cat_n$. In the *ID* part, id_i is the ID of the component corresponding to cat_i. Meanwhile, the three next parts contain information of how the graph (G) can be constructed: G is formed by adding q graphs of q components corresponding to $id_{i1}, id_{i2}, ..., id_{iq}$ (in "Nodes of Graph" part) into a single graph G. Information in "Edges of Graph" is used to create edges in G: there are directed edges from each head of the graph corresponding to id_{ej1} to a head of the graph corresponding to id_{ej2} that labeled by $label_j$ (with $j\ from\ 1\ to\ t$). Heads of G are heads of the graphs corresponding to $id_{h1}, id_{h2}, ..., id_{hk}$. Here, *heads* of graph are main semantic concepts of the graph. For example, in the graph of "the big cat", "cat" is the main semantic concept, so it is the head of graph. Heads of graphs for Noun Phrase, Verb Phrase, Adjective Phrase, Adverb Phrase, are normally Nouns, Verbs, Adjectives and Adverbs respectively. A graph may has more than one head, for example, graph for "dog and cat" has two heads including "dog" and "cat".
To formalize this graph building step, *we propose an algorithm as in Algorithm 1.* Based on a set of rules, graphs for components in the pattern are constructed then combined into a single graph.

Algorithm 1 Building graph for a semantic role using a syntactic pattern

```
GRAPH=EMPTY
Find Rule L that matches the given syntactic pattern
if the right side of L is empty then
  if "Nodes of graph" part of L is empty then
    return EMPTY
  else
    if there is one element in "Nodes of graph" of L and it is equal to $id_1$ then
      return GRAPH = a graph has only 1 node represented by the text
      content of the component corresponding to $id_1$ (it is also the head of the
      graph)
    else
      return EMPTY
    end if
  end if
end if
if the right side of the pattern has one or more components then
  for j from 1 to q do
    Apply Algorithm 1 to the component corresponding to $id_{ij}$ and add the
    retrieved graph into GRAPH
  end for
  for j from 1 to t do
    In GRAPH, create directed edges labeled by $label_j$ from Head of subgraph
    corresponding to $id_{ej1}$ to Head of subgraph corresponding to $id_{ej2}$
  end for
  for j from 1 to k do
    heads of the graph corresponding to graphs of $id_{hj}$ are assigned as heads
    of the current GRAPH
  end for
end if
return GRAPH
```

For example, to build graph for "the big cat" matching the pattern $NP >$ $DET\ JJ\ NN$, with IDs are the same as syntactic categories, we define 3 rules:
Rule 1: For adjective, the graph contains only 1 node labeled by the given adjective.
Pattern: $JJ >$
ID: $JJ >$
Heads of Graph: JJ
Nodes of Graph: JJ
Edges of Graph:
Rule 2: For common noun, the graph contains only 1 node labeled by the given noun.
Pattern: $NN >$
ID: $NN >$
Heads of Graph: NN
Nodes of Graph: NN
Edges of Graph:
Rule 3: For NP, there are two nodes in graph corresponding to the adjective and the noun. An edge labeled by "attribute" connects the noun to the adjective.
Pattern: $NP > DET\ JJ\ NN$
ID: $NP > DET\ JJ\ NN$
Heads of Graph: NN
Nodes of Graph: JJ, NN
Edges of Graph: $NN - JJ - attribute$
By using the algorithm, "the big cat" has the pattern matching Rule 3: "the big cat", "the", "big", "cat" correspond to NP, DET, JJ and NN respectively. In order to build graph for "the big cat", we have to build graph for the components corresponding to JJ, NN (see *Nodes of Graph* part) which are "big" and "cat". "big" has pattern of $JJ >$ that matches Rule 1. By using Rule 1, the graph B for "big" has only one node labeled by "big" and this node is also the head of B. Similarly, the graph C for "cat" also has only one node labeled by "cat" which is the head. Afterward, we add these two graphs into a single one, then add an edge labeled by "attribute" from head of graph C to the head of graph B. Thus, we get the graph as in Figure 3.3 and "cat" is the head of this graph.
Some other examples of how to build graph for syntax patterns are as follows:
$PP > IN\ NP$ ("in the house"): The graph is the same as the graph of the NP ("the house").
$VP > V\ TO\ NP$ ("go to Paris"): There is one node in graph representing the V ("go"), and then the graph of the NP ("Paris") is added to this graph. There is one relation from the V to the head of the graph representing the NP. The V

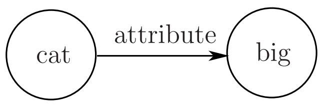


Figure 3.3: A graph for the semantic role corresponding to "the big cat"

is the head of this graph.

3.4.2 Graph for semantic frame

After building graphs for all semantic roles in a frame, a graph can be constructed for the frame as follows:
First of all, we create one node labeled by the name of the frame and call this type of node "Frame node".
Next, for the target of the frame, a node labeled by its text content is created. This node has the type of "Target node". We also add an directed edge from Frame node to Target node labeled by "frame_target".
Next, we build graph G_i for each semantic role $Role_i$ in the frame and add all of the obtained graphs into the current graph. The type of all nodes in G_i is called "Word node".
Finally, the "Target node" is connected to each Head of G_i by an directed edge labeled by the name of semantic role corresponding to G_i.
For example, given the frame Beat_Opponent which has "beat" as Target, "the big cat" and "the small dog" correspond to two semantic roles named by "winner" and "loser": We first add a node labeled by "Beat_Opponent", then a node labeled by "beat" to the graph. Then, the two graphs for "the big cat" and "the small dog" obtained by using algorithm in section 3.4.1 are added to the current graph. The final graph represented the frame is as in Figure 3.4.

3.5 Graph completion

After building graphs for a set of frames detected from text, we perform the next step which is connecting frame graphs together to form graph representing the whole text.
In this section, we consider the text of "The small dog is beaten by the big cat. The smart mouse beats the cat. Peter likes dog." as an example. *We call the text "Animal".* There are three frames in the text:
"Beat_Opponent"(target: "beat", winner: "the big cat", loser: "the small dog");

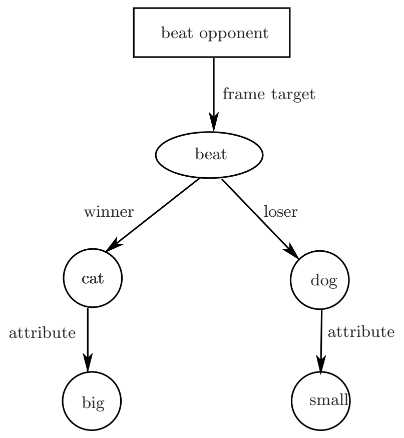


Figure 3.4: Graph for frame Beat_Opponent

"Beat_Opponent"(target: "beat", winner: "the smart mouse", loser: "the cat");
"Experiencer_Focus"(target: "like", experiencer: "Peter", content: "dog").
After the previous step (Graph for semantic frame), we obtain graphs representing frames in text. They are inputs for this Graph completion step which is defined as follows:
First of all, we combine all frame graphs into a single graph G. In the example of the text "Animal", an unconnected graph as in Figure 3.5 is obtained.

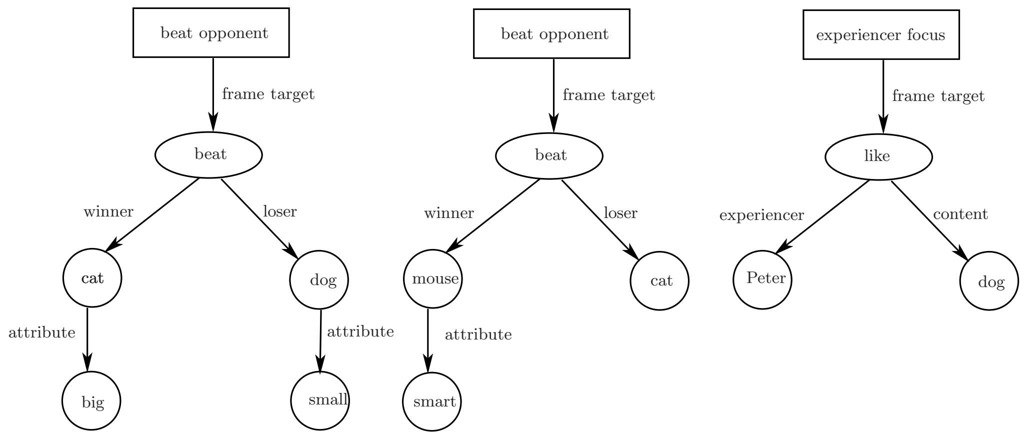


Figure 3.5: "Animal" example: three frame graphs

Next, taken into account the fact that there may be more than one instance of one target in the whole text (in the example of “Animal” text, there are two instances of target “beat” linked to frame “Beat_Opponent”), for each frame, we create a new node that has the same label as the target node but its type is called “Target Instance node”. This type of node is used to represent the occurrence of words that are targets in frame. For each node in the frame graph linked to the “Target node” by an edge e, we link it to the “Target Instance node” by the same direction, same label as e, then e is removed. A directed edge labeled by “target instance” from the “Target node” to “Target Instance node” is also added to the graph. In our example, the graph G has the new form as in Figure 3.6.

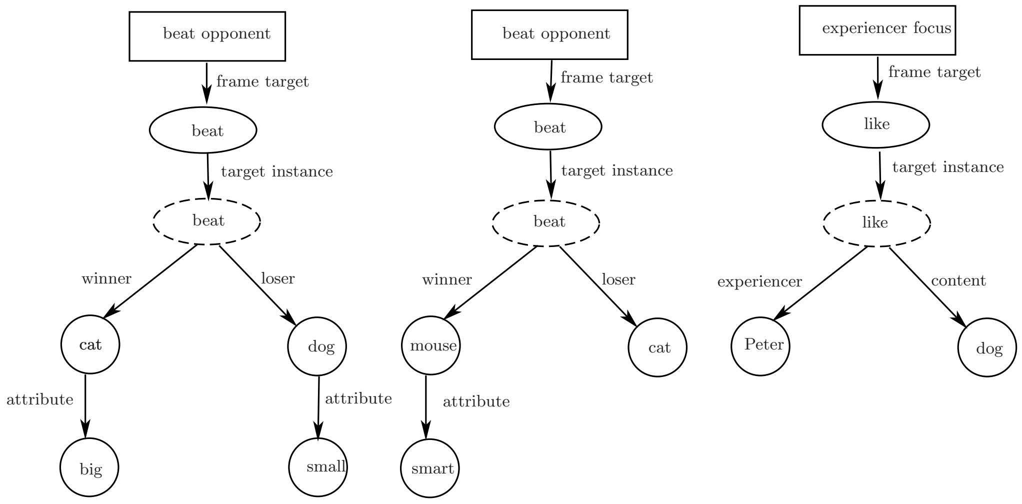


Figure 3.6: “Animal” example: Three frame graphs with Target Instance nodes

Next, for nodes of types “Frame node” or “Target node”, we merge all nodes with same type and same label into a single node. We also merge all edges with the same source, same target and same label for these two types of nodes. In our example, after this merging task, the graph for “Animal” text is an un-connected graph as in Figure 3.7.
Finally, relations between frame graphs may be added to the graph. A simple way to add relations between frames is using *frame position* in text. For the set of frames detected from the text, we consider the set of their targets: if frame F1 has the target that immediately precedes the target of frame F2, then we create an directed edge from the Frame node of F1 to the Frame node of F2 labeled by “next_frame”. *Another way* is to create a node called “TOP” which is connected to all frame nodes by directed edges with labels “frame”. The graph in this case is as in Figure 3.8.

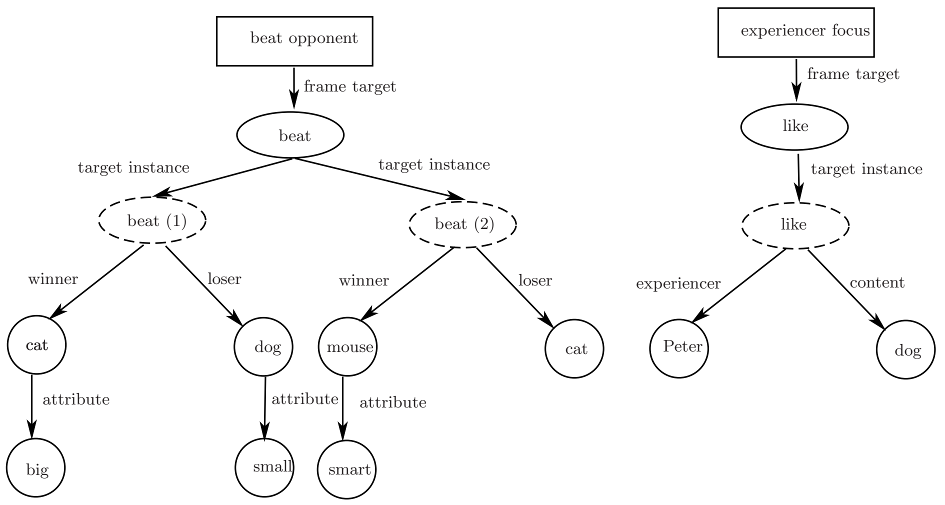


Figure 3.7: “Animal” example: Graph after merging task

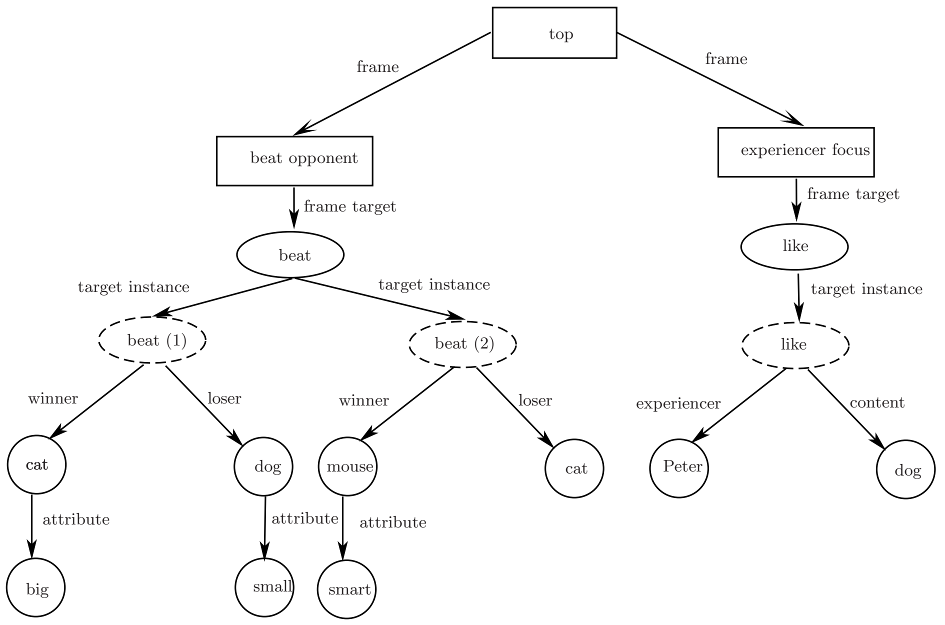


Figure 3.8: “Animal” example: Completed graph with TOP

3.6 Some other processes

3.6.1 FrameNet Relations between frames

FrameNet has a frame ontology which contains information about relations between frames. It also provides information about relations between frame elements. By using this ontology, we can enrich our graph with relations between frames. Given a graph containing a set of frames, we first extract the frame-to-frame relations of the frames in the graph from FrameNet frame ontology. Then for each type of relation, we apply some steps to add related information into the current graph.
There are several frame relations in FrameNet such as "Inheritance", "Perspective on", "Subframe" (see Section 1.5). We resolve each type individually.

"Inheritance" relation

This is the strongest relation between two frames, corresponding to is-a relation in many ontologies. With this relation, anything which is strictly true about the semantics of the parent frame must correspond to an equal or more specific fact in the child frame. For instance, frame "Quitting_a_place" is inherited from "Departing". The frame element "Co_theme" in "Departing" frame must correspond to frame element "Co_participant" in "Quitting_a_place" frame. To add inheritance relations to graph, if a child frame exists in our graph, we add the parent frame (if it has not existed in the graph) to graph by creating a "Frame node" labeled by Parent frame's name, a "Target node" and a "Target Instance" labeled by the same as "Target node" and "Target Instance node" of the child frame. The edges between these three nodes are the same as in the child frame. In the next step, we connect the child frame node to the parent frame node by an directed edge labeled by "parent frame". After that, for nodes connected to the child frame's Target Instance node by the semantic roles that have corresponding roles in the parent frame, we connect the corresponding parent frame's Target Instance node to them by edges labeled by the corresponding roles in the parent frame. Figure 3.9 shows an example of how to include inheritance in FGMT graph. The text in this example is "Peter quitted".

"Perspective on" relation

Given a frame F which provides a particular perspective on a parent frame P. If F exists in our graph, we may add P to graph in the same way as Inheritance relation, but the edge label between F and P in the graph is changed to "perspective on". Moreover, we may consider the relation information of P in FrameNet to find

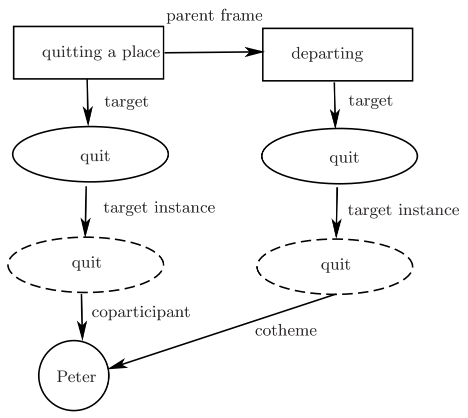


Figure 3.9: Graph with inheritance relations

all other frames that also provides a particular perspective on P. Then, these frames are added to the graph in a similar way but with respect to frame P. For example, consider the sentence “Peter hired Mary”. “hire” is the lexical unit evoking the frame “Hiring”, “Mary” is the semantic role “Employee” and “Peter” is the semantic role “Employer”. This frame perspectivizes the “Employment Start” frame: “Employer” and “Employee” in “Hiring” correspond to the semantic roles with the same names as in “Employment Start”. Similarly, frame “Get a job” is also perspectivizes the “Employment” with the same semantic role relations as “Hiring”. The graphs for the text before and after adding “Perspective on” relation are as in Figure 3.10.

“Subframe” relation

If the subframe is exist in our graph, the complex frame is added to the current graph in a similar way to Inheritance relation.

Conclusion

We propose to use these kinds of relations for a further semantic analysis of text or some applications like smart information retrieval, question answering etc. For the example, in Figure 3.10, the graph before adding “Perspective on” relation shows that we have only a “hiring” scenario (someone hires employee). But, after adding the relation, we know that there are also two other scenarios: an “employment start” scenario (an employment starts) and a “getting a job” scenario (someone gets a job). This information is quite interesting and it can be used for a further

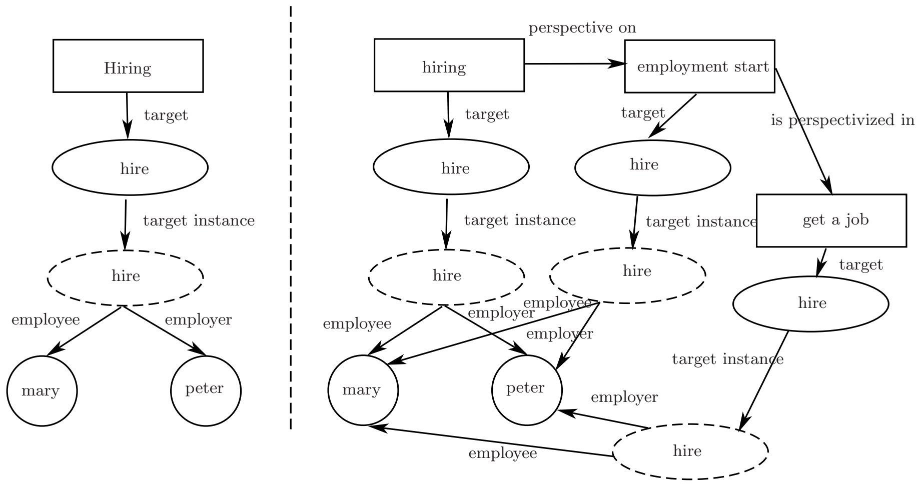


Figure 3.10: Graph without and with "perspective on" relations from FrameNet

semantic analysis application.

3.6.2 Removing non-important frames, words

One problem need to be considered is that the number of nodes and frames may be very large in a text. Furthermore, shallow semantic analysis step may return some frames that are not related to our domain or not so important. To overcome this problem, we can use the methods as follows:
First, for words, a list of stopwords or a vocabulary of the given domain can be used.
Second, for frames, we can also define "stop-frames" for the working domain including frames have not much meaning in our domain.

3.6.3 Weighted graph

We may assign weight scores to nodes and edges of graph by using frequency or normalized frequency or any other suitable type of weighting score.

3.6.4 Change the form of graphs

Instead of using graph with labeled edges, for each edge, we can create a node represent the label and then two unlabeled and undirected edges from the source node to the label node and from the label node to the target node. This form is closer to the form of Conceptual Graphs. For example, graph of "the big cat" can be represented as in Figure 3.11.

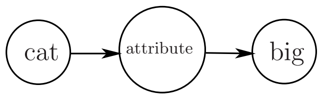


Figure 3.11: Example of graph without labeled edges

Another type of graph can be built is undirected graph: Instead of using directed edges, we use un-directed edges connecting nodes in graph.

3.7 Hybrid Model based on FGMT

After creating FrameNet-based Graph Model for Text, frequent subgraph mining (see section 2.1.2) can be applied to create a Hybrid model (see Section 2.3) based on our FGMT.
As to frequent subgraph mining method, we use *gSpan* (Graph-based Substructure Pattern mining, Yan and Han, 2002, [26]) which is a method that discovers frequent substructures without candidate generation. This method builds a new lexicographic order among graphs, and maps each graph to an unique minimum depth-first search code as its canonical label. Based on this lexicographic order, gSpan adopts the depth-first search strategy to mine frequent connected subgraphs efficiently.
A Hybrid model based on FGMT can be built as follows:
Given a text corpus $C = \{d_1, d_2, ..., d_n\}$ as input, we *first* represent each of text d_i by a graph G_i in FGMT. A graph corpus $G = \{G_1, G_2, ..., G_n\}$ is obtained after this step.
Next, we apply gSpan algorithm to the graph corpus G and retrieve a set of subgraphs $S = \{g_1, g_2, ..., g_m\}$. All of these subgraphs are chosen to be terms in the hybrid model.
Finally, d_i is represented by a vector $v_i = \{vg_{i1}, vg_{i2}, ..., vg_{im}\}$ where $vg_{ij} = 1$ if g_j is a subgraph of G_i and $vg_{ij} = 0$ if g_j is not a subgraph of G_i (Boolean weighting).
For example, we consider a corpus containing 3 texts $C = \{d_1, d_2, d_3\}$. Each text d_i is first represented by graph G_i with $i = \overline{1,3}$. Next, by using gSpan with threshold = 60% on the three graphs $\{G_1, G_2, G_3\}$, we retrieve (for instance) 4 subgraphs g_1, g_2, g_3, g_4 where g_1 is subgraphs of G_1,G_2; g_2 is subgraphs of G_2,G_3; g_3 is subgraphs of G_1,G_3 and g_4 is subgraphs of G_1,G_2. Term set includes all of obtained subgraphs: $T = \{g_1, g_2, g_3, g_4\}$. Each text can represented as a vector in the matrix of the Hybrid model: d_1 is described by $\{1, 0, 1, 1\}$ because g_1, g_3, g_4

are subgraphs of G_1, but g_2 is not. Similarly, d_2, d_3 are described by $\{1, 1, 0, 1\}$, $\{0, 1, 1, 0\}$, respectively. The Hybrid model can be viewed as the following matrix:

$$\left(\begin{array}{ccccc} & g_1 & g_2 & g_3 & g_4 \\ d_1 & 1 & 0 & 1 & 1 \\ d_2 & 1 & 1 & 0 & 1 \\ d_3 & 0 & 1 & 1 & 0 \end{array}\right)$$

The Hybrid Model based on FGMT can be applied directly in most model-based classification algorithms like Naive Bayes, Decision Tree etc. When we apply this hybrid model on classification, the shallow semantic information (semantic roles) contained in FGMT is used indirectly.

3.8 Applications

3.8.1 Text classification

In our Hybrid Model based on FGMT, a text is represented simply by a vector of Boolean values. Therefore, this model can be applied in most model-based classification algorithms. We will evaluate its effectiveness in Chapter 5.

3.8.2 Pattern mining

As mentioned in Sections 2.1.2 and 3.7, a frequent subgraph mining algorithm like gSpan (see Section 3.7) can be applied on a graph corpus in FGMT to mine interesting patterns. In fact, our FGMT describes a type of semantic frame structure with the relations between frame elements, targets and frames, therefore, the patterns mined may tell us the interesting information about frame structure in the data. Moreover, because our frames describe the semantic scenes (see Sections 1.4,1.5), so the pattern mining result also contains semantic information.

3.9 Some issues

3.9.1 Limits of linguistic resources

In order to detect frames and identify semantic roles, we need linguistics resources which provide frame definitions, semantic role definitions, lexical unit definitions and examples. However, it is difficult to find resources that contain all the frames and lexical units we need on our working domain. In the newest version of FrameNet (1.5), the number of frames and lexical units are 1029 and 11829, respectively, but it is not a large number when working with a large text corpus. Moreover, in FrameNet, there is a limited number of domains and almost of them are still at a general level. For example, Communication domain with frames Candidness, Commitment, Conversation etc.; Emotion domain with frames Emotion_Active, Experiencer_obj, Experiencer_subj etc; Health domain with frames Cure, Recovery etc. To solve this problem, one solution could be building a resource that can provide enough data for the working domain. In fact, it is not easy for a huge domain but for a specific domain we believe that it is realizable.

3.9.2 Effectiveness of semantic parsing/semantic role labeling task

Semantic role labeling system have been shown to perform reasonably well in some controlled experiments, with F1 measures in the low 80s on standard test collections for English [17]. However, when applying these methods on our model, the effectiveness of this number (low 80s) needs to be examined.

3.9.3 Building rule set to construct graph for syntactic constituent

Section 3.4.1 showed that building rule set to construct graph for syntactic constituent (correspond to a semantic role) is an important task in our model. However, it is not a simple task, because in order to obtain a graph describe semantics about a syntactic constituent, it is necessary to have a deep study and experiment about the relation between syntax components.

3.9.4 Some linguistics problems of FGMT graphs

In this version of FGMT, we do not deal with discourse problems like pronouns, tenses in text, relations between discourses, anaphora etc. For example, in anaphora problem, FGMT does not support to detect the a referential tie to some other linguistic entity in the same text: Given the text “Mary likes Peter. However, he does not like her.”, it is impossible to refer “she” to “Mary” and “he” to “Peter” in this version of FGMT.

Chapter 4

IMPLEMENTATION

In this chapter, we present our implementation of a tool, which takes a text corpus as an input and supports the whole processes of building FrameNet-based Graph Model for Texts (see chapter 3), Graph Models for Web Documents (GMWDs) (see section 2.2) and Hybrid models (see Sections 2.3, 3.7). This tool is used for the experiments described in chapter 5.

4.1 System overview

Figure 4.1 shows an overview of our system which contains *four main modules*: (1) *Shallow Semantic Analysis*: This module consists of two components including Preprocessing and Interface to Shalmaneser. Its function is to identify semantic frames together with their frame elements. (2) *Graph Builder*: We use this module to build graphs representing texts. (3) *Frequent Subgraph Mining*: It is used to discover frequent subgraphs in a graph corpus by using gSpan (see Section 3.7). (4) *Data Exporter*: Output data is exported by this module.
The *goals* of the system are to build FGMT, GMWDs and to apply Hybrid representation techniques on these graph models for a text corpus.
Our system takes *a corpus of categorized texts as input*, then performs as the following steps:
First, Preprocessing function analyzes the text input by using part of speech tagging, named entity recognition and syntax parsing.
Second, output of Preprocessing function is either sent to Shalmaneser (for FGMT) or to Graph builder (for GMWDs). In the Shallow Semantic Analysis module, there is an interface which allows the interaction between our tool and Shalmaneser. It sends the output data of Preprocessing function to Shalmaneser, and then receives the result of Shalmaneser. Afterward, the result is sent to Graph

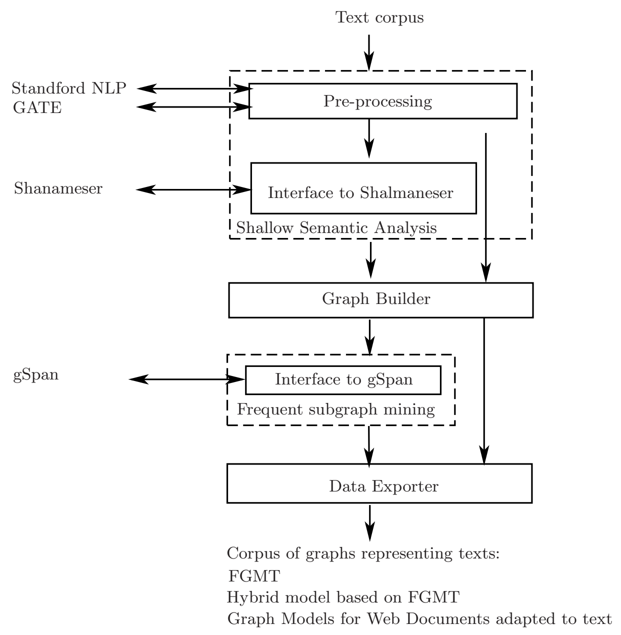


Figure 4.1: Overview of implementation

builder.
Third, Graph builder module constructs graph representation for texts in the given corpus. This module supports our FrameNet-based Graph Model and GMWDs. After this step, we may send the output data either to Frequent subgraph mining module which has an interface to gSpan (see Section 3.7) to run frequent subgraph mining in case of building hybrid representation model or directly to Data exporter.
At final step, Data exporter module is used to export the results of graph representation models or hybrid graph representation models for all texts in the corpus.

4.2 Preprocessing

At this step, the input text is analyzed by using Part of speech tagging, Named Entity Recognition and Syntax tagging. Currently, we use GATE[1] and Standford CoreNLP[2] for this step.

[1]http://gate.ac.uk/
[2]http://nlp.stanford.edu/software/

4.3 Interface to Shalmaneser

An interface to interact with Shalmaneser was implemented. There are two options to use Shalmaneser: using both preprocessing and machine learning functions of Shalmaneser or using our own preprocessing and Shalmaneser's machine learning functions (see Section 3.3).

4.4 Graph Builder

This module provides functions to construct graph representation of texts in FGMT with different options for the relation between frame nodes: no relation, simple relation and FrameNet relation (see Section 3.5 and 3.6.1).
Other types of graph supported in this tool are Graph Models for Web Documents (see Section 2.2). GATE is used to analyze text, and then we retrieve a list of tokens, named entities (organisation, location, time, person, money etc.) and co-location (two or more words that often go together). We use these above types to build nodes in graphs and support all types of GMWDs: Standard, Simple, n-Distance, n-Simple Distance, Absolute Frequency and Relative Frequency (see Section 2.2).
In Graph Builder, we use jGraphT[3] which is a free Java Graph Library to represent graphs. Also, FrameNet and Wordnet are the two linguistic resources used to extract frame relations and word information for the graph construction.

4.5 Frequent subgraph mining

In the case of Hybrid models, after building graphs representing all the texts using Graph Builder, we send graphs to gSpan, then receive its result to construct a hybrid model.

4.6 Data Exporter

We can either export data in FrameNet-based Graph Model, Graph Models for Web Documents or Hybrid models.
In FGMT and GMWDs, data is exported in TEXT or DOT format. TEXT format data contains text-encoded graphs of all the text:
"t # N" means the Nth graph,
"v M L" means that the Mth vertex in this graph has label L,

[3]http://jgrapht.org/

"e P Q L" means that there is an edge connecting the Pth vertex with the Qth vertex. The edge has label L.
M, N, P, Q, and L are integers.
There are also text files that maps from integers value to string value of labels. This format is suitable for gSpan tool.
Another data format in FGMT is DOT which is a plain text graph description language. Each of text in corpus is represented by a single DOT file.
For hybrid models, the data is exported in CSV format as the followings example:

	$Term\ 1$	$Term\ 2$	$Term\ 3$	...	$Term\ n$	$Category$
$Text\ 1$	1	0	1	...	1	$Cat\ 1$
				...		
$Text\ m$	0	1	1	...	1	$Cat\ n$

Chapter 5

EXPERIMENTS

5.1 Introduction

The aims of the experiments presented in this chapter are to test the feasibility of FGMT, to identify problems that might happen as one uses the model in practice, and to evaluate the effectiveness of the model on several text analysis and text mining tasks.
For these *first* experiments of FGMT, a small text corpus, which consists of 110 short texts collected from Wikipedia[1], was selected as the experimenting data.
The experiments were performed as follows:
First of all, we built FGMT and Hybrid model based on FGMT (see Chapter 3) for a text corpus.
Next, we evaluated the obtained Hybrid models on two text mining tasks including supervised text classification (Decision Tree, K-Nearest Neighbours, LibSVM), and unsupervised text classification (K-Means, Hierarchical Clusterer). In order to have the first comparison between FGMT and other models, we also implemented a typical Vector Space Model and a Simple method in Graph Model for Web Documents (see Chapter 4), and tested them in the same classification algorithms and data.
The implementation described in Chapter 4 was used for these experiments.

5.2 Data

Our experimenting data was a text corpus (DATA A) including 110 short biographies of famous people. The texts were collected from Wikipedia: 55 Chief

[1]http://en.wikipedia.org/wiki/

Feature	**Value**
Average number of sentences	7
Average number of tokens per sentence	22 (punctuation included)
Syntax	Varies from short sentence to long sentence. Lot of sentences with complex syntax. Some sentences are in ill-formed syntax.

Table 5.1: Data information

executive officers (CEO) and 55 football players (some data samples can be found in Appendix A). Each text includes personal information like Name, Birthday, Home country, Job, Career, Salary etc. Table 5.1 shows detail information about the data.
We also used two others datasets created by dividing DATA A into two sets: 73% for training (40 texts each category): DATA B and 27% for testing (15 texts each category): DATA C.

5.3 Building FGMT

Input: Set of texts
Output: Set of graphs representing texts in FGMT model
As mentioned in Chapter 3, the *first* important part of FGMT construction is Shallow Semantic Analysis which identifies semantic frames from a text using FrameNet resource and several supervised machine learning algorithms. Shallow Semantic Analysis needs a set of frame-annotated sentences as training data to annotate frames for an unannotated data. Because of the limited coverage of FrameNet, when building FGMT, we should first evaluate the effectiveness of using FrameNet as training data in our working domain. Afterward, if the result is not close to what we expect, one possible way is to extend the frame data of FrameNet, and then to develop a new training data that is suitable for the working domain.
Dealing with the above problem, in Shallow Semantic Analysis step, Shalmaneser was tested with two options: (1) Option 1 - using FrameNet 1.5 as training data and DATA A as testing data. (2) Option 2 - using DATA B as training data and DATA C as testing data.

Then, in the *second* part, the testing result of the option obtaining better accuracy in Shallow Semantic Analysis was chosen to build graphs in FGMT.

5.3.1 Shallow Semantic Analysis task with Option 1

In this task, we evaluated the effectiveness of Shallow Semantic Analysis using FrameNet 1.5 as training data (see Section 3.3).
Experiment Setups
Training data: annotated sentences extracted from FrameNet 1.5
Testing data: DATA A (unannotated)
Results

Data	**Total number of frames**	**Total number of distinct frames**
CEO	1891	211
Football Player	2277	219
CEO + Football Player	4168	302

Table 5.2: Frames detected by Shalmaneser using FrameNet 1.5 as training data

Shalmaneser detected a number of frames related to our domain like "Being_born", "Leadership", "Membership","First_rank", "Being_employed", "Business" , "Award" etc. However, by checking the results manually, we observed that the results *contained a great deal of errors*. In many frames, it could detect only targets but no semantic role. One of the possible reasons is the difference between the training data collected from FrameNet and the testing data. Moreover, some meaningless frames such as "Locative_relation", "Aggregate", "Time_vector", "Temporal_collocation" etc. occurred many times in the results. Also, there were some incorrect frame assignments like "Wearing" for target "in". In contrast, Shalmaneser did not detect several types of important information in the texts because they are not defined in FrameNet, such as salary information, transfer fee information, etc. There are several possible reasons of these problems, for instance, the limited size, domain, coverage of FrameNet etc. In Table 5.2, we show information about the frames detected by Shalmaneser. From the table, we can see that the total frames that can detected in our whole data corpus by using FrameNet 1.5 as training data, is 4168 with 302 distinct frames.

5.3.2 Shallow Semantic Analysis task with Option 2

Because of the low accuracy of Shallow Semantic Analysis using FrameNet 1.5 as training data (as mentioned in Section 5.3.1), we defined another Shallow Semantic Analysis scheme using DATA B as training data, and DATA C as testing data. There were two steps here: (1) Preparing training data: First, DATA B was annotated by using Shalmaneser with FrameNet 1.5 as training data. Then, we corrected the retrieved annotated data manually. We also removed several useless frames and added some new frames and frame elements that are meaningful in our domain. (2) Testing: we use this training data to identify frames in the testing data (unannotated version of DATA C).

Preparing Training Data (DATA B)

First, from the annotation result of testing on DATA B using Shalmaneser with FrameNet as training data, some important information types for our domain but did not appear in the result was determined. Next, we defined our own frames that represent those information types in FrameNet format. Some examples of new frames are "Compensation_package", "Work_responsibility", "Salary", "Award" etc. For instance, the frame "Salary" contains "salary" as target, "employee", "employer", and "money" as semantic roles. Furthermore, there is one case that we need to add a new semantic role to an exist FrameNet frame: semantic role "Fee" to frame "Transfer". Based on the new frame set, we corrected the annotation result of DATA B obtained by using Shalmaneser with FrameNet as training data. At this step, SALTO (A. Burchardt et al., 2006 [2]) was used as annotation tool. SALTO is a tool for manual annotation within an intuitive, easy to use graphical environment. It is developed for the annotation of semantic roles and semantic classes in the FrameNet paradigm. Moreover, SALTO can work with the SALSA/TIGER XML which is the input/output format of Shalmaneser. By using this tool, we can define a list of frames in FrameNet format and annotate text manually.
The information about frames in our DATA B after manual correction and annotation can be found in Table 5.3. From the table, we can see that the total numbers of frames and distinct frames in our data decrease from 4168 to 1913 and from 302 to 73, respectively. A large number of meaningless or irrelevant frames were removed. For more detail information about frames in our data after manual annotation correction, see Appendix B.

Data	**Total number of frames**	**Total number of distinct frames**
CEO	1108	64
Football Player	805	37
CEO + Football Player	1913	73

Table 5.3: Frames information after annotation correction

Testing on DATA C

We used the annotated version of DATA B (obtained in the previous step) as training data to identify semantic frames along with their semantic roles for the unannotated version of DATA C. Then, the results were evaluated by using the manual annotated version of DATA C (we also annotated DATA C in the same scheme as DATA B) as gold standard. Here, there are two tasks of machine learning algorithms need to be evaluated: Frame Disambiguator and Role Assignment System (see Section 3.3).
Result
Frame Disambiguator: Accuracy - 87.23%
Role Assignment System: Accuracy - 96.23%
Although the overall accuracy results of both tasks are more than 87%, the precision and recall scores are not good for some pairs of (target,frame). There are some possible reasons for this problem such as the quality of data (syntax problem of Wikipedia texts), the quantity of training data (small size, several testing cases do not occur in training data) and the performance of Shalmaneser. Some detail results of this experiment can be found in Appendix C. By checking the result, we observed that using this Option 2 is much better than using FrameNet as training data because the training and testing sets in this option are close to each other. Thus, we decided to use the Shallow Semantic Analysis results of Option 2 which contains 30 frame-annotated texts of DATA C as input for the next experiments.

5.3.3 Building graphs for DATA C

We built graphs in FGMT for all texts in DATA C by using the results of the previous step (see Section 5.3.2), and then we obtained 30 graphs for 30 texts (15 CEOs and 15 football players). The relations between frames used in this experiments was "position relations" (see Section 3.5). Some examples of FGMT graphs are shown in the Appendix D.

5.4 Building "Hybrid Model based on FGMT" for DATA C

After building FGMT for DATA C, we used gSpan and the implementation described in chapter 4 to create Hybrid models based on the FGMT graphs (see Section 3.6) obtained in the previous experiment (see Section 5.3.3). The output data was exported in CSV format (see Section 4.5) which can be applied directly in some normal machine learning softwares like WEKA[2], KNIME[3], MALLET[4] etc. for text classification.

5.5 Text classification

In this experiment, we evaluate effectiveness of "Hybrid models based on FGMT" on several classification algorithms. As an approach towards a comparison between Hybrid models based on FGMT and another models, a typical Vector Space Model and Hybrid models based on our implementation of the Simple Graph Models for Web Documents (SGMWD)(see Section 2.2 and Chapter 4) were also tested.

Supervised Text classification
- Given: A pre-defined category set (CEO, Football Player), A set of categorized texts as training data
- Input: A set of un-categorized texts
- Task, Output: To predict category for each input text
- Algorithms: Decision Tree (J48), k-NN, and LibSVM
- Tool: Weka
- Test mode: Cross-Validation 5 folds, in which the original sample is randomly partitioned into k sub-samples. Of the k sub-samples, a single sub-sample is retained as the validation data for testing the model, and the remaining (k 1) sub-samples are used as training data. The cross-validation process is then repeated k times (the folds), with each of the k sub-samples used exactly once as the validation data. The k results from the folds then can be averaged (or otherwise combined) to produce a single estimation.

Unsupervised Text classification
- Given: NONE
- Input: A set of uncategorized texts

[2]www.cs.waikato.ac.nz/ml/weka/
[3]www.knime.org/
[4]mallet.cs.umass.edu/

- Task, Output: To group input texts into clusters
- Algorithms: KMeans, Hierarchical Clusterer (HC)
- Tool: Weka
- Test mode: We ignore the labels of original sample, then group data into 2 groups, then during the test phase it assigns classes to the clusters, based on the majority value of the class attribute within each cluster, then accuracy is reported.

Data
- 3 "Hybrid models based on FGMT" for unannotated DATA C obtained in Section 5.4, with gSpan frequency threshold = 10%, 15% and 20%
- 3 Vector Space Model (word stems as terms and tf-idf score as weighting method) for unannotated DATA C with frequent threshold (for terms selection, see Section 1.1) = 0%,10% and 20%
- 3 Hybrid models based on our implementation of the Simple Graph Model for Web Documents (SGMWD) (see Section 2.2 and Chapter 4) for unannotated DATA C, with gSpan frequency threshold = 10%, 15% and 20%

Results
The highest accuracy for each type of model in each method is shown in Table 5.4:

Model type	**J48**	**k-NN**	**LibSVM**	**k-Means**	**HC**
Hybrid Model based on FGMT	96.667%	100%	100%	100%	73.333%
Vector Space Model	86.667%	96.667%	100%	56.667%	53.333%
Hybrid Model based on SGMWD	83.333%	96.667%	96.667%	66.667%	60.000%

Table 5.4: Results of text classification

Observations
Referring to the result table, a significant improvement is seen in unsupervised methods (K-Means and HC) by using hybrid approaches versus the Vector Space Model. Especially, the hybrid models based on FGMT get the best results in all methods. Our models improve the results of K-Means, HC and J48 by at least 10%.

5.6 Discussion

The experiments in this chapter show that in order to apply FGMT in practice, we should first analyze the effectiveness of FrameNet in our working domain. In case FrameNet does not support our domain well, one possible solution is to extend the frame data of FrameNet and to develop our own training data.
By using the same experiment setups for VSM, SGMWD, and FGMT, we can see that the hybrid methods have significant outperformance in unsupervised text classification. Furthermore, the Hybrid models based on FGMT shows the best results on some supervised classification experiments.
However, only a very small data was used to experiment, so we consider these experiments as only "the first approach" towards a comparison between FGMT and other models in text mining tasks. It needs some other experiments on larger data and different domains to evaluate the effectiveness of FGMT.

Chapter 6

Conclusion and Future Works

In conclusion, our contributions are summarized as follows:
First, we propose a FrameNet-based Graph Model for Text which is a graph model that captures structural and shallow semantic information of texts. A graph in FGMT provides a picture about semantic frames, targets and semantic roles in a text based on Frame Semantics theory and FrameNet linguistic resource. Based on this FGMT, a hybrid model can be built by using frequent subgraph mining tool, and then it can be applied directly in most machine learning algorithms.
Second, a tool building FGMT and Hybrid models based on FGMT for a corpus of texts and exporting data for text classification was implemented.
Third, we presented several approaches to apply our FGMT in text classification and frequent pattern mining.
Finally, by using this tool, we performed some experiments testing the *feasibility* of our FGMT on a small corpus. These experiments addresses several issues when applying FGMT in practice due to the limits of FrameNet. Furthermore, in some other experiments, we *evaluated the effectiveness of Hybrid models based on FGMT on several text classification algorithms*. As an approach towards a comparison between FGMT and other models, some Vector Space Models and Hybrid Models based on our implementation of Simple Graph Model for Web Documents were also tested in the same algorithms. It is interesting that the experiment results of our Hybrid models based on FGMT *surpass significantly the traditional VSM* in all unsupervised text classification algorithms that were tested.
This thesis is just the first step to build a complete model. In fact, FGMT can capture more semantics than Vector Space Model, but the contained semantic information is still shallow. In order to capture the complete semantics of the whole text, we need to deal with many problems of discourse like pronouns, tenses, discourse relations, anaphora etc. and they can be considered as our future work.

Moreover, improving the effectiveness of semantic role labeling systems, building rule set for graph construction etc., performing more experiments are also need to be considered. As to application, one possible future work for FGMT is “synthesis of a collection of texts”. Given a collection of texts, in which each text is represented by a graph, we compute the “least common subgraph” which can be the basis for the text synthesis process.

Abbreviation

CGs	= Conceptual Graphs.
FE	= Frame Element.
FEs	= Frame Elements.
FGMT	= FrameNet-based Graph Model for Text.
FSM	= Frequent Subgraph Mining.
GMWDs	= Graph Models for Web Documents.
HC	= Hierarchical Clusterer.
LUs	= Lexical Units.
SGMWD	= Simple Graph Model for Web Documents.
SRs	= Semantic Roles.
VSM	= Vector Space Model.

Appendix A

Data Samples

Sample text of CEO:
Steve Jobs
Steve Jobs (February 24, 1955 October 5, 2011) was an American businessman, designer and inventor. He is best known as the co-founder, chairman, and chief executive officer of Apple Inc. Through Apple, he was widely recognized as a charismatic pioneer of the personal computer revolution and for his influential career in the computer and consumer electronics fields. Jobs also co-founded and served as chief executive of Pixar Animation Studios. He became a member of the board of directors of The Walt Disney Company in 2006, when Disney acquired Pixar.

Sample text of Football Player:
Roberto Baggio
Roberto Baggio (born 18 February 1967) is a retired Italian footballer. He is widely regarded as one of the finest footballers of all time (4th at a Fifa internet poll; member of the Fifa World Cup Dream Team). Baggio won both the Ballon d'Or and the FIFA World Player of the Year in 1993. He is the only Italian player ever scoring in three World Cups. He is also one of the top 5 all-time goalscorers for Italy. Baggio is known as Il Divin Codino (The Divine Ponytail), for the hairstyle he wore for most of his career and his Buddhist background.

Appendix B

Frame Information

Being_born	People_by_vocation	Age	Award
Name_conferral	Becoming	Footballer_Position	First_rank
Holding	Regard	Leadership	Win_prize
Assistance	Appointing	Membership	Awareness
Capability	Appearance_football_match	Sign_agreement	Transfer
Becoming_a_member	Motion	Activity_start	Commerce_buy
Competition	Play_for_team	Occupy_rank	Success_or_failure
Inclusion	Statement	Take_place_of	Activity_finish
Being_employed	Possession	Being_named	Commerce_pay
			Quitting

Table B.1: Frames in football players data after manual correction

Leadership	People_by_vocation	Appointing	Businesses
Education_teaching	Residence	Food	Being_born
Childhood	Becoming	Sport	Education_degree
Fields	Being_employed	Motion	Change_of_leadership
Membership	Manufacturing	Quitting	Name_conferral
Intentionally_create	Award	Aggregate	First_rank
Occupy_rank	Holding	Text_creation	Text
Statement	Work_Reponsibility	Inclusion	Being_named
Activity_start	Cogitation	Age	Promotion
Achieving_first	Awareness	Research	Cure
Removing	Creating	Travel	Employing
Regard	Hiring	Possession	Process_end
Activity_finish	Coming_to_believe	Experiencer_focus	Performers_and_roles
Personal_relationship	Cause_change_of_position_on_a_scale	Commerce_sell	Change_position_on_a_scale
Assistance	Desiring	Judgment	

Table B.2: Frames in CEO data after manual correction

Appendix C

Semantic Analysis experiments

Precision = Number of found instances that are correct / Number of found instances Recall = Number of found instances / Number of expected instances F-Score = 2 * Precision * Recall / (Precision + Recall)

Target,Frame	**Precision**	**Recall**	**F-Score**
Chief!NNP, Leadership	1.0000 (7/7)	1.0000 (7/7)	1.0000
President!NNP, Leadership	1.0000 (10/10)	1.0000 (10/10)	1.0000
bear!VBN, Being_born	1.0000 (24/24)	1.0000 (24/24)	1.0000
play!VBD, Competition	0.7143 (5/7)	0.8333 (5/6)	0.7692
win!VBD, Win_prize	1.0000 (14/14)	1.0000 (14/14)	1.0000

Table C.1: Some detailed results of Frame Disambiguator

Semantic Role	**Precision**	**Recall**	**F-Score**
Target,Frame: JJS_best, First_rank			
Attribute	0.8750 (7/8)	0.8750 (7/8)	0.8750
Item	0.0000 (0/1)	0.0000 (0/2)	0.0000
Limits_of_consideration	0.0000 (0/1)	0.0000 (0/0)	0.0000
NONE	0.9799 (195/199)	0.9898 (195/197)	0.9848
Target,Frame: NNP_Chairman, Leadership			
Governed	0.7500 (3/4)	0.6000 (3/5)	0.6667
Leader	0.5000 (3/6)	0.6000 (3/5)	0.5455
NONE	0.9774 (259/265)	0.9811 (259/264)	0.9792
Role	0.8000 (4/5)	0.8000 (4/5)	0.8000
Time	0.0000 (0/0)	0.0000 (0/1)	0.0000
Target,Frame: NNP_Chief, Leadership			
Duration	0.0000 (0/2)	0.0000 (0/0)	0.0000
Governed	1.0000 (2/2)	0.6667 (2/3)	0.8000
Leader	0.2500 (1/4)	0.3333 (1/3)	0.2857
NONE	0.9776 (262/268)	0.9740 (262/269)	0.9758
Role	0.7500 (6/8)	1.0000 (6/6)	0.8571
Time	0.0000 (0/0)	0.0000 (0/3)	0.0000

Table C.2: Some detailed results of Role assignment system

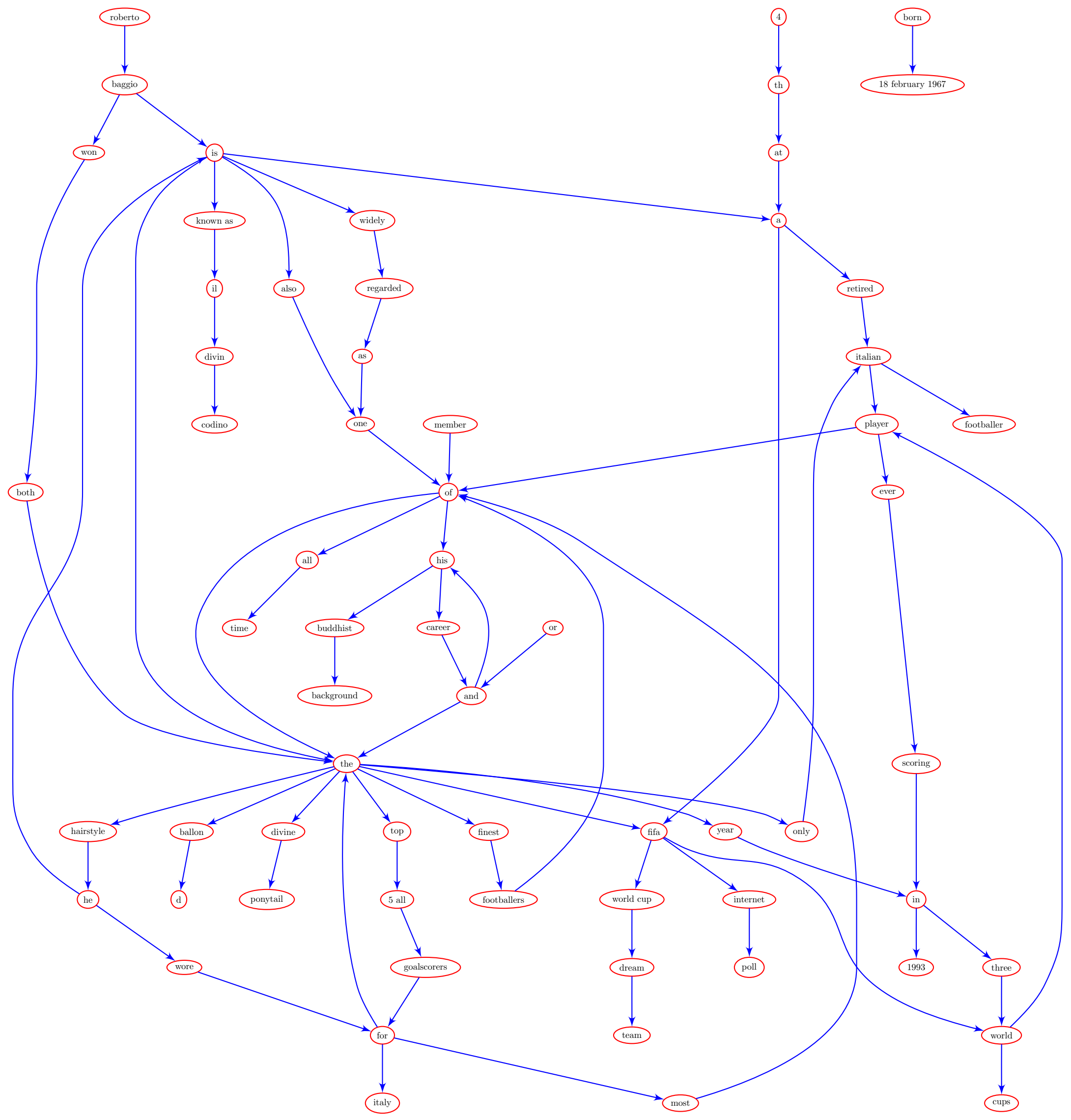
Appendix D

Graphs Samples

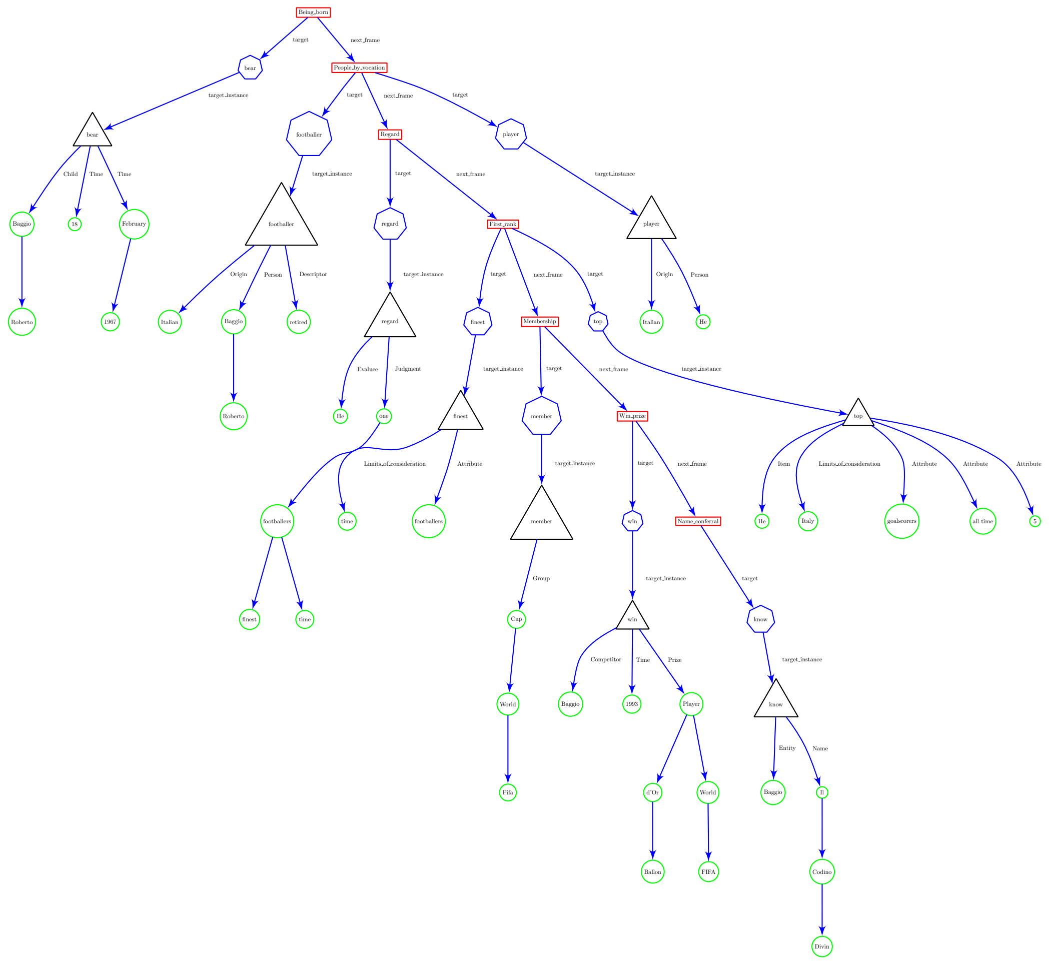
In our FGMT graphs:
rectang: frame node
polygon: target node
triangle: target instance node
circle: word node

Simple Graph Models for Web Documents of ”Roberto Baggio”.

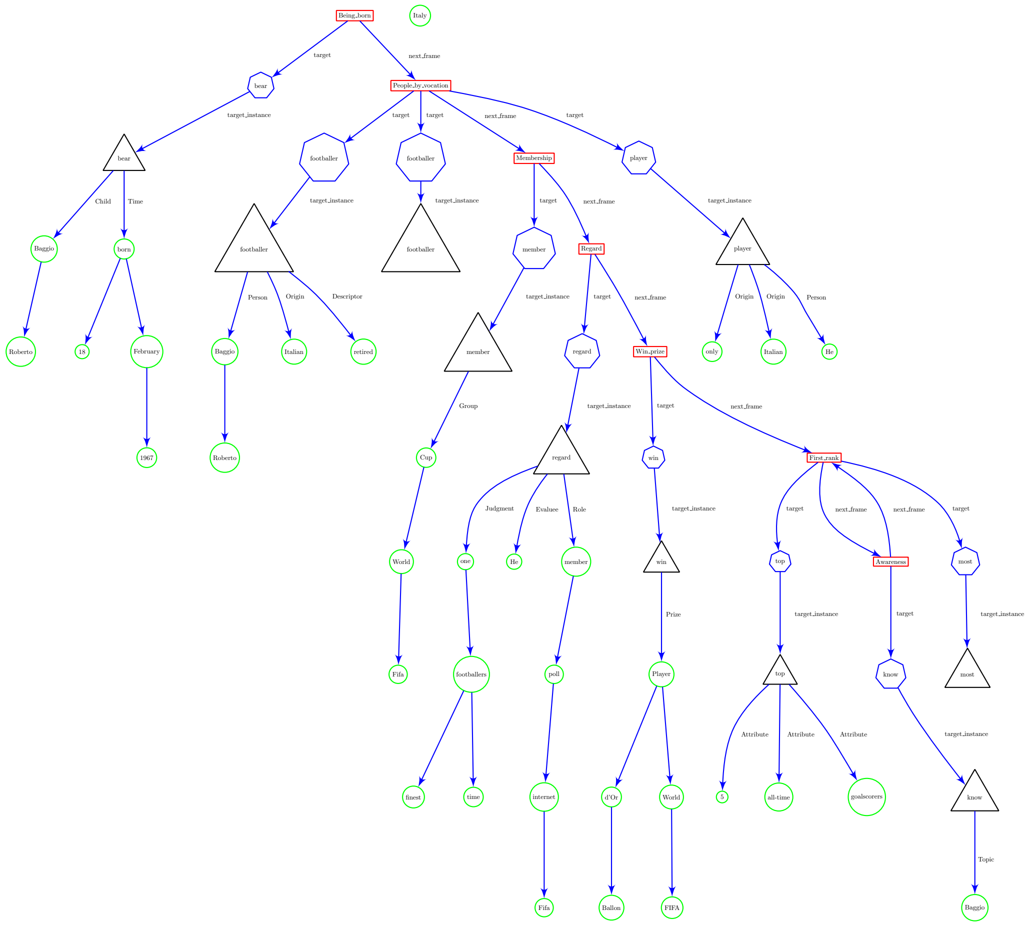
There are some nodes that do not connected to others because of punctuation/syntax problems in given plain text.



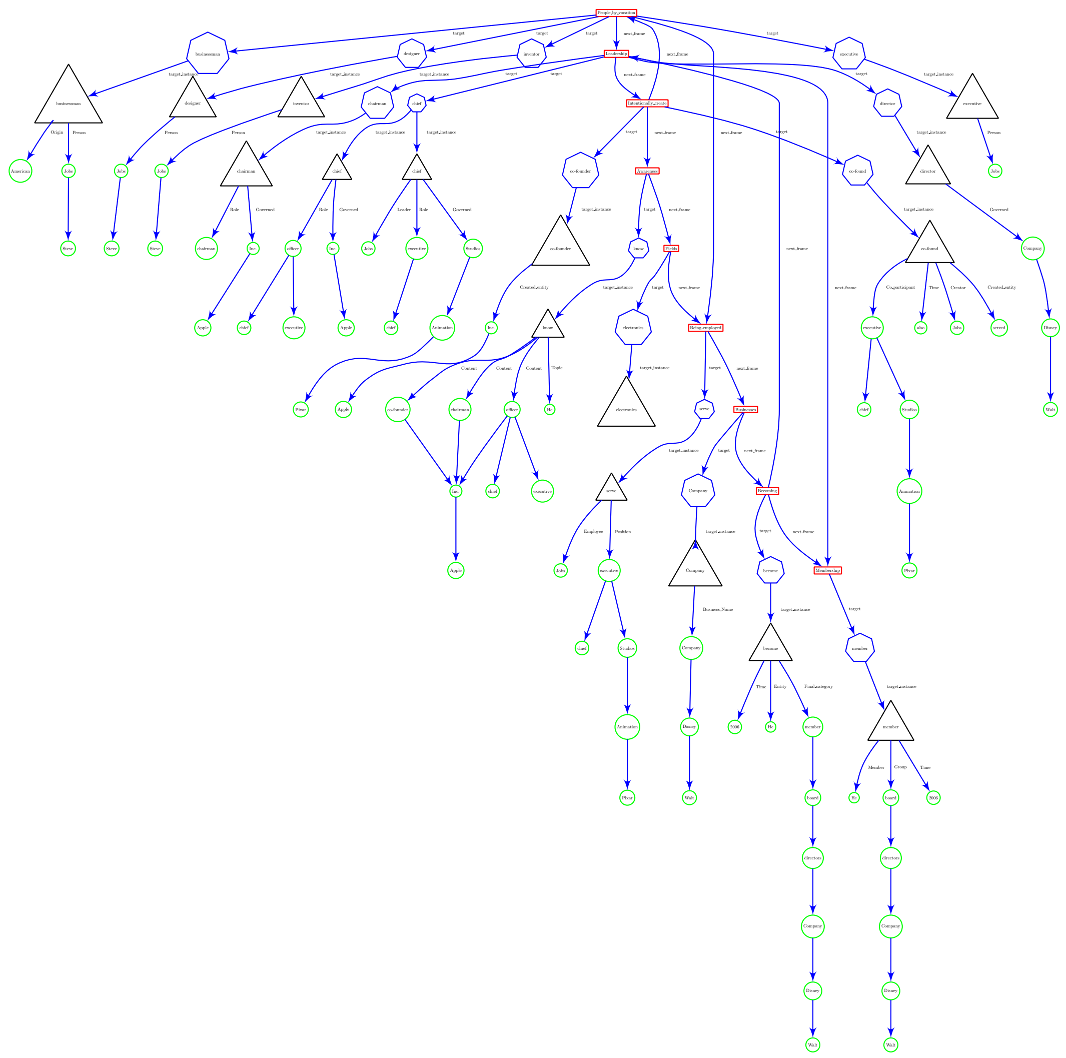
Expected FGMT graph of a football player - Roberto Baggio (using manual annotation)



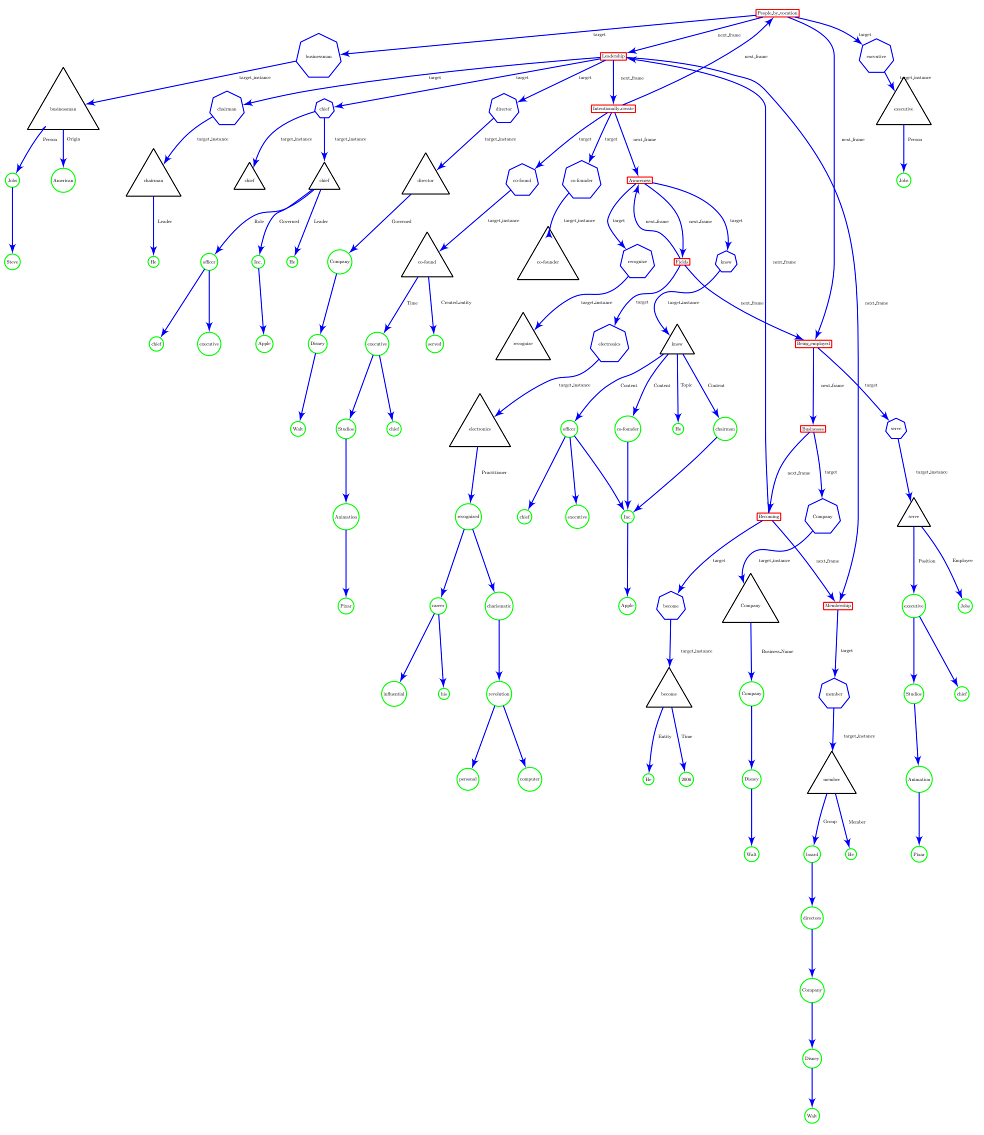
Obtained FGMT graph of a football player - Roberto Baggio (using our tools).



Expected FGMT graph of a CEO - Steve Jobs



Obtained FGMT graph of a CEO - Steve Jobs



Bibliography

[1] Collin F. Baker, Charles J. Fillmore, and John B. Lowe. The berkeley framenet project. In *COLING-ACL*, pages 86–90, 1998.

[2] Aljoscha Burchardt, Katrin Erk, Anette Frank, Andrea Kowalski, and Sebastian Pado. SALTO: A versatile multi-level annotation tool. In *Proceedings of LREC-2006*, Genoa, Italy, 2006.

[3] Katrin Erk and Sebastian Pad. Shalmaneser a toolchain for shallow semantic parsing. *Computational Linguistics*, 6(2), 2006.

[4] Katrin Erk and Sebastian Pado. A powerful and versatile xml format for representing role-semantic annotation. In *Proceedings of LREC-2004*, Lisbon, 2004.

[5] D. Gildea and D. Jurafsky. Automatic labeling of semantic roles, 2002.

[6] Fritz Hamm. Frame semantics. In *The Cambridge Encyclopedia of the Language Sciences*. Cambridge University Press, 2009.

[7] Svetlana Hensman and John Dunnion. Constructing conceptual graphs using linguistic resources. In *Proceedings of the 4th WSEAS International Conference on Telecommunications and Informatics*, TELE-INFO'05, pages 34:1–34:6, Stevens Point, Wisconsin, USA, 2005. World Scientific and Engineering Academy and Society (WSEAS).

[8] Andreas Hotho, Andreas Nrnberger, and Gerhard Paa. A brief survey of text mining. *LDV Forum - GLDV Journal for Computational Linguistics and Language Technology*, 2005.

[9] Akihiro Inokuchi, Takashi Washio, and Hiroshi Motoda. An apriori-based algorithm for mining frequent substructures from graph data. In *Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery*, PKDD '00, pages 13–23, London, UK, UK, 2000. Springer-Verlag.

[10] C. Jiang, F. Coenen, R. Sanderson, M. Zito, F. Coenen, R. Sanderson, and M. Zito. Text classification using graph mining-based feature extraction. *Knowledge-Based Systems*, 23(4):302–308, 2010.

[11] Chuntao Jiang, Frans Coenen, and Michele Zito. *Matrix*, 00(0):1–31, 2008.

[12] T. Meyyappan K.Lakshmi. A comparative study of frequent subgraph mining algorithms. *International Journal of Information Technology Convergence and Services*, 2(2), 2012.

[13] Michihiro Kuramochi and George Karypis. Frequent subgraph discovery. In *Proceedings of the 2001 IEEE International Conference on Data Mining*, ICDM '01, pages 313–320, Washington, DC, USA, 2001. IEEE Computer Society.

[14] A. Markov and M. Last. Model-based classification of web documents represented by graphs. In *In Proc. of WebKDD 2006: KDD Workshop on Web Mining and Web Usage Analysis, in conjunction with the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2006*, 2006.

[15] A. Markov, M. Last, and A. Kandel. Fast categorization of web documents represented by graphs. In *Proceedings of the 8th Knowledge discovery on the web international conference on Advances in web mining and web usage analysis*, WebKDD'06, pages 56–71, Berlin, Heidelberg, 2007. Springer-Verlag.

[16] A. Markov, M. Last, and A. Kandel. The hybrid representation model for web document classification. *Int. J. Intell. Syst.*, 23(6):654–679, June 2008.

[17] Lluís Màrquez, Xavier Carreras, Kenneth C. Litkowski, and Suzanne Stevenson. Semantic role labeling: an introduction to the special issue. *Comput. Linguist.*, 34(2):145–159, June 2008.

[18] Sonia Ordoñez Salinas and Alexander Gelbukh. Information retrieval with a simplified conceptual graph-like representation. In *Proceedings of the 9th Mexican international conference on Advances in artificial intelligence: Part I*, MICAI'10, pages 92–104, Berlin, Heidelberg, 2010. Springer-Verlag.

[19] Martha Palmer, Daniel Gildea, and Paul Kingsbury. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106, March 2005.

[20] Josef Ruppenhofer, Michael Ellsworth, Miriam R.L. Petruck, Christopher R. Johnson, and Jan Scheffczyk. *FrameNet II: Extended Theory and Practice.* International Computer Science Institute, Berkeley, California, 2006. Distributed with the FrameNet data.

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

[22] A. Schenker, H. Bunke, M. Last, and A. Kandel. *Graph-theoretic techniques for Web content mining. Series in Machine P erception and Articial Intelligence.* World Scientic, 2005.

[23] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Comput. Surv.*, 34(1):1–47, March 2002.

[24] John F. Sowa. *Conceptual Structures: Information Processing in Mind and Machine.* Addison-Wesley, 1984.

[25] John Wang, editor. *Encyclopedia of Data Warehousing and Mining, Second Edition (4 Volumes).* IGI Global, 2009.

[26] Xifeng Yan and Jiawei Han. gspan: Graph-based substructure pattern mining. In *Proceedings of the 2002 IEEE International Conference on Data Mining*, ICDM '02, pages 721–, Washington, DC, USA, 2002. IEEE Computer Society.