VERBOSE LABELS FOR SEMANTIC ROLES

by

Ravikiran Vadlapudi

B.Tech., International Institute of Information Technology, 2008

M.Sc., International Institute of Information Technology, 2010

Thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Science

in the
School of Computing Science
Faculty of Applied Sciences

SIMON FRASER UNIVERSITY
Spring 2013

APPROVAL

Name: Ravikiran Vadlapudi

Degree: Master of Science

Title of Thesis: Verbose Labels for Semantic Roles

Examining Committee: Dr. Oliver Schulte
Chair

Dr. Anoop Sarkar, Senior Supervisor
Associate Professor, Computing Science
Simon Fraser University

Dr. Fred Popowich, Supervisor
Professor, Computing Science
Simon Fraser University

Dr. John Dill, SFU Examiner
Professor Emeritus,
School of Interactive Arts and Technology
Simon Fraser University

Date Approved:

Abstract

We introduce a new task that takes the output of semantic role labeling and associates each of the argument slots for a predicate with a verbose description such as *buyer* or *thing_bought* to semantic role labels such as 'Arg0' and 'Arg1' for predicate like "buy". Ambiguous verb senses and syntactic alternations make this a challenging task. We adapt the frame information for each verb in the PropBank to create our training data. We propose various baseline methods and more informed models which can identify such verbose labels with 95.2% accuracy if the semantic roles have already been correctly identified. We extend our work to text visualization to illustrate the importance of verbose labeling. As a proof of concept, we built an interactive browser for human history articles from Wikipedia, called lensingWikipedia.

Keywords: Semantic Role Labeling; Verbose Labeling; Text Visualization; Verb Sense Prediction; PropBank

To my family!

Language is a process of free creation; its laws and principles are fixed, but the manner in which the principles of generation are used is free and infinitely varied. Even the interpretation and use of words involves a process of free creation.

NOAM CHOMSKY

Acknowledgments

First, I would like to express my very great appreciation to Dr. Anoop Sarkar for his valuable and constructive suggestions during this research work. Working with him has been a wonderful experience. I would like to thank my committee, Dr. Fred Popowich, Dr. John Dill and Dr. Oliver Schulte, for their valuable feedback. I would also like to extend my thanks to Dr. Christopher Collins for introducing me to the area of data visualization and for his feedback on this work.

Many thanks to my lab-mates particularly, Baskaran, Marzieh, Maryam, Majid, Rohit, Max, Porus, Ann, David and Milan for their help and support. I would also like to thank my friends Raj, Rajesh, Siddharth, Vivek, Udit, Samrat and Pradeep for making my stay in Vancouver a memorable one.

Finally, I wish to thank my family for their love and support throughout my study.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Task Description

Semantic Role Labeling (SRL) is a task of identifying semantic arguments for a verb of a sentence and defining their roles. It aims to identify "who" did "what" to "whom, where and how" structure in a sentence. The predicate (typically a verb) establishes "what" took place and other constituents in the sentence filling in as participants, also called arguments. The following is an example of SRL:
[Agent The boy] [V_{pred} hit] [Recipient a ball].

The primary task of SRL is to define the relationship between the predicate and its arguments where the relation is drawn from a defined set of relations applicable for that predicate. Relations such as *Agent* and *Recipient* are called *Semantic roles* and the task of automatically generating these is called *Semantic Role Labeling* (SRL). The relation set heavily influences the extent of semantic analysis with more fine-grained relations edging towards deeper semantic analysis of language text. For example, *Agent* and *Recipient* roles can be replaced with:
[Hitter The boy] [V_{pred} hit] [Thing_hit a ball].

Roles such as *Hitter* and *Thing_hit* are called as Verbose labels. In this thesis, we define the task of assigning verbose labels to semantic roles as Verbose Label Prediction.

Verbose label prediction is closely related to verb sense identification (Ye and Baldwin, 2006) but it is not exactly the same task. For instance, in: "Sony bought official rights for the Steve Jobs movie" given that 'Sony' is Agent for 'bought', a verbose label *buyer* can be assigned. The verbose labels for arguments depend on the sense of the predicate, c.f.

"Imports have gone down" and "Portfolio managers go after the highest rates" which show two different senses of the predicate 'go'. 'Imports' and 'highest rates' are both assigned a Patient semantic role. 'Imports' is assigned a verbose label *an entity in motion* and 'highest rates' is assigned a verbose label *goal*. Verbose labels do not always change with the predicate sense, e.g. in "John made up all the answers on his midterm" and "Loews Corp makes Kent cigarettes", even though the sense of the predicate 'make' is different in both instances the same set of verbose labels can be assigned to the arguments, so that 'John' and 'Loews Corp' can be defined as *creator*. One might share verbose labels across many different predicate senses and so the verbose label generation task cannot be equated to verb sense identification even though sense identification is very important for this task. *Verbose Label Prediction*, which (as far as we know) has not been a direct subject of a detailed experimental study before (although some SRL systems (Koomen et al., 2005) pick a default verbose label in their web-based SRL tool).

In this thesis, we aim to automatically produce verbose and easy to understand descriptions in natural language for semantic roles that vary according to predicate and argument. We extend our work on semantic parsing to text visualization mainly to illustrate the importance of verbose labeling. As a proof of concept, we built an interactive browser for human history articles from Wikipedia, called lensingwikipedia (`http://www.lensingwikipedia.cs.sfu.ca`).

1.2 Motivation

There are several application areas of semantic parsing that would benefit from having verbose labels for the semantic roles. In question answering (QA), the verbose labels can match user queries, e.g. asking questions about a *buyer* or *seller* is possible since the QA system can match user queries about these more specific entities using verbose labels. Searching for information extraction applications can also benefit from having verbose labels since it expands the domain of what can be searched. Finally, our major motivation for this work was a text visualization and visual analytics system we built that uses SRL output and displays the information in text using the semantic roles. In our visualization it was crucial to convert the abstract semantic roles to the verbose labels that are easy to navigate and use by the users of the visualization application. Our visualization tool is described in detail in chapter 4.

1.3 Semantic Role Labeling

Semantic Role Labeling (SRL) has become a standard shallow semantic parsing task thanks to the availability of annotated corpora such as the Proposition Bank (PropBank) (Palmer, Gildea, and Kingsbury, 2005) and FrameNet (Fillmore, Wooters, and Baker, 2001). The following examples shows SRL annotation in the PropBank notation:

1. [Arg0 Ports of Call Inc.] reached agreements to [V_{pred} sell] [Arg1 its remaining seven aircraft] [Arg2 to buyers that weren't disclosed] .

2. [Arg0 Bell Industries Inc.] [V_{pred} increased] [Arg1 its quarterly] [Arg4 to 10 cents] [Arg3 from seven cents] .

3. [Arg1 Bond prices] [AM-DIS also] [V_{pred} edged] [Arg5 higher] .

The semantic roles for a predicate are numbered sequentially from Arg0 to Arg5 where Arg0 is assigned to argument acting as an Agent, Arg1 to argument acting as a Patient or Theme and so on. In addition to these, a category of adjunct semantic roles is defined with the tag ArgM along with 13 functional tags denoting the role of the element, such as ArgM-TMP (temporal markers) and ArgM-LOC (locatives markers).

Penn Treebank (Marcus, Marcinkiewicz, and Santorini, 1993) is a corpus of naturally-occurring sentences annotated with their linguistic structures showing syntactic and semantic information. Figure 1.1 shows a syntactic tree from the Penn Treebank. For SRL, this syntactic tree representation of a sentence is linearized into a sequence of its syntactic constituents and each constituent is assigned to one of several semantic roles using the linguistic context of constituent token. PropBank is created in a similar manner to the Penn Treebank on a verb-by-verb basis. For each of the predicate, a sample of sentences from the corpus are grouped into one or more major sense, depending on syntactic behavior which correlates with different types of allowable arguments, and each major sense turns into a single sub-categorization frame. The predicate *accused* in Figure 1.1 takes frame *accuse.01* with *A1* 'he' defined as *accused* and *A2* 'of conducting illegal ..' defined as *crime.*

Since linguistic information from syntactic trees is essential for SRL, syntactic parsing plays an important role of automatically generating syntactic trees similar to the Penn Treebank trees. For this purpose, a *parser* learns grammar rules automatically from the Penn Treebank and generates a syntactic parse tree for a given sentence. With the availability of such parsers (Charniak, 2000; Collins, 2003) considerable research has gone into SRL including several shared tasks (Carreras and Màrquez, 2004; Carreras and Màrquez,

```
(S (NP-SBJ-1 he)
  (VP was
    (VP accused
      (NP-3 *-1)
      (PP-CLR of
          (S-NOM (NP-SBJ *-3)
              (VP (VP conducting
                    (NP illegal business))
                and
                (VP possessing
                    (NP illegal materials))))))))
```

Figure 1.1: An example of syntactic tree from Penn Treebank

2005; Surdeanu et al., 2008). In CoNLL-2004 (Carreras and Màrquez, 2004) and CoNLL-2005 (Carreras and Màrquez, 2005), the task was, given a sentence and syntactic information in the form of a parse tree, for each of the target verb in the sentence, semantic role constituents for that verb have to be recognized. In CoNLL-2004, SRL systems were expected to use only partial parsing information. In CoNLL-2005, with the availability of full syntactic parsing tools, performance of the SRL systems greatly improved. This shared task resulted in some of the state-of-the-art performing systems(`http://aclweb.org/aclwiki/index.php?title=Semantic_Role_Labeling_(State_of_the_art)`) and we give a brief overview of some of them [1].

The system in Koomen et al. (Koomen et al., 2005)(p/r/f%: 82.82/76.78/79.44) achieved the best performance accuracy in CONLL-2005 shared task. It takes the output of multiple argument classifiers and combines them into a coherent predicate-argument output by solving an optimization problem which takes into account the classifier recommendations as well as a set of constraints like there is only one Arg0 for a predicate and so on. It reports a significant improvement in overall SRL performance through this inference. The second best performing system was proposed in Toutanova et al. (Toutanova, Haghighi, and Manning, 2005)(p/r/f%: 81.90/78.81/80.32). In this work, instead of using the best full syntactic parse from the Charniak parser they use the top 10 parses. The third best system in the shared task was proposed in Pradhan et al. (Pradhan et al., 2004) (p/r/f%: 82.95/74.78/78.63). It uses features derived from different syntactic views, and combines them within a phrase-based chunking paradigm. Syntactic features are derived from the

[1] System descriptions are summarized from abstracts of respective papers

Charniak parser (Charniak, 2000) and the Collins parser (Collins, 2003) and some features are derived from constituents assigned as semantic roles by Support Vector Machine (SVM) classifiers.

Another system among the best performing systems was proposed in Liu & Sarkar (Liu and Sarkar, 2007) (p/r/f%: 72.3/65/68.5) which inspired our implementation of SRL system. This work uses a variant of the LTAG formalism called LTAG-spinal and the associated LTAG-spinal Treebank (Shen, 2006). On treating multiple LTAG derivation trees as latent features, they achieve state-of-the-art performance. Another interesting work which deviates from the previous attempts at SRL was proposed in Vickrey & Koller (Vickrey and Koller, 2008). This work addresses the issue about sparsity of predicate-argument path features in a syntactic parse tree. In this method, they generate canonical forms of an input sentence using hand-crafted rules and perform SRL on them achieving a close to state-of-the-art accuracy(f%: 78).

All these systems efficiently and accurately label constituents with abstract semantic roles. However, once we annotate a large dataset with SRL output, navigating and searching the annotations is a cumbersome task, since the only search keys are these abstract labels like Arg0. In this thesis, we introduce a new task that takes the output of semantic role labeling and associates each of the argument slots for a predicate with a verbose description. We call this task as Verbose Label Prediction. In this thesis we define different solutions to the verbose label prediction problem, and show that the task can be done with high accuracy when given accurate SRL information.

1.4 Applications to Information Visualization

Semantic parsing with verbose labels finds its application in areas such as question answering, where the task is to provide information salient to the user's needs. In this thesis we illustrate advantages of verbose labels in the field of Information Visualization where the information is available as unstructured text. Information visualization techniques aim to translate abstract information into a visual form to get new insights and have been successfully developed for a wide range of tasks. However, text visualization remains challenging, mostly due to the complex underlying grammatical structure of natural language.

Visually highlighting key terms and relations among them to gain new insights is a common practice. Tools such as JigSaw (Stasko, Görg, and Liu, 2008) allow the intelligence analysts to examine the relationships among entities mentioned in a document collection. Another common practise in text visualization is topic modeling. Clustering documents into classes, based on words, with each of the cluster referring to some topic and visualizing them as clusters is provided in tools like In-Spire (Wong et al., 2004). Both the techniques assume text as a bag-of-words. The downside of such technique is it completely ignores linguistic structures. WordsEye (Coyne and Sproat, 2001) is one of the few tools which uses NLP algorithms for visualization of text. WordsEye is a system for automatically converting text into representative 3D scenes which uses a syntactic parser to derive the dependencies of entities in the context of a sentence. This area of text visualization is called as Scene Generation which is entirely different from the kind of text visualization attempted in this thesis. The term *text visualization*, in this thesis, is used from the perspective of data mining using visualization through a search interface allowing an analyst to browse collections of text articles.

For searching and browsing text articles, Faceted search has proven to be very effective (Hearst and Stoica, 2009). Faceted search, also called faceted navigation or faceted browsing, is a technique for accessing information organized according to a faceted classification system, allowing users to explore a collection of information by applying multiple filters. A faceted classification system classifies each of the information element along multiple explicit dimensions, enabling the classifications to be accessed and ordered in multiple ways rather than in a single, pre-determined, taxonomic order. The facets correspond to properties of the information elements. They are often derived by analysis of the text of an item using entity extraction techniques or from pre-existing fields in a database such as author, descriptor, language, and format. Faceted browsing has been applied for closed domain datasets like Nobel prize winners, and recipes (Stoica and Hearst, 2007) and for digital libraries and e-commerce shopping websites.

To allow analysts to search for entities related to specific events requires a sophisticated language analysis. For this purpose, state-of-the-art Natural Language Processing (NLP) algorithms can be used to identify both entities and relationships among such entities in a piece of text. Visualization can then show underlying entity relationships enabling discovery of the hidden information. But (to our knowledge) no attempt has been made in this direction. In this thesis, we extend our work (described in Chapter 3) on semantic parsing to text visualization mainly to illustrate the importance of verbose labeling. As a proof of concept, we built an interactive browser for human history articles from Wikipedia, called lensingwikipedia (`http://www.lensingwikipedia.cs.sfu.ca`).

1.5 Contributions

Two main contributions of this thesis are:

1. We introduce a new task called Verbose Label Prediction that takes the output of semantic role labeling and associates each of the argument slots for a predicate with a verbose description. We define different solutions to the verbose label prediction problem, and show that the task can be done with high accuracy when given accurate SRL information.

2. We extend our work on semantic parsing to text visualization to illustrate the importance of verbose labeling as well as the importance of NLP algorithms in text visualization. As a proof of concept, we built two novel interactive browsers for human history articles from Wikipedia.

1.6 Outline

The remainder of this thesis is structured as follows.
Chapter 2 presents our implementation of SRL tool focusing mainly on engineering aspects and the performance evaluations.
Chapter 3 provides more details about the verbose label prediction problem and presents different solutions to this problem and their performance evaluations.
Chapter 4 presents two novel interactive browsers for human history articles which is an application of the verbose labeling problem.
Chapter 5 concludes by summarizing the thesis and providing future directions of research in this area.

Chapter 2

Semantic Role Labeling

2.1 Introduction

Semantic Role Labeling (SRL) is a process of identifying who, did what, to whom, where and how in a sentence. For example, in the sentence *John kicked a ball*, *kick* is the action, also called as a **predicate**, and *John* and *ball* are actors, also called as **arguments**, of the predicate where *John* plays the role of a *kicker* and *the ball* plays the role of a *thing kicked*. SRL aims to identify all arguments for each predicate in a sentence and to assign pre-defined roles to its arguments. Hence the steps involved in SRL are: predicate identification, argument identification and argument classification.

Argument Identification

The task of argument identification is defined as, given a predicate, selecting all argument candidates of the predicate. Hence argument identification is defined as a binary classification task and all of the arguments discarded are not considered for argument classification.

Argument Classification

The task of argument classification is defined as, given a predicate and identified argument candidates, assigning pre-defined roles to each of the arguments. The pre-defined roles can be verbose labels (*kicker*) or can be abstract labels A0-A4 and AM, as defined in the PropBank (Palmer, Gildea, and Kingsbury, 2005). Hence argument classification is defined as a multi-class classification task where the classes are the pre-defined role labels.

In this chapter, we describe our implementation of the SRL tool focusing on the engineering aspects of the tool. Section 2.2 gives a brief introduction to LTAG-Spinal formalism which is used for training our model. Section 2.3 explains the underlying architecture of the tool and the alternatives of SRL pipeline that we used in our tests. Certain design decisions taken during the building process are also covered in this section. In section 2.4, we list the features used in the system followed by the experimental evaluations in section 2.5.

2.2 LTAG-spinal Treebank

LTAG-Spinal formalism is a variant of Lexicalized Tree Adjoining Grammar (LTAG). Similar to LTAG, there are two types of elementary trees (e-tree), namely initial and auxiliary trees. Both of the trees are in spinal form. A spinal initial tree is composed of a lexical spine from the root to the anchor and a spinal auxiliary tree is composed of a lexical spine and a recursive spine from the root to the foot node.For example, in Figure 2.1 (figure taken from (Liu and Sarkar, 2009)), the lexical spine for the auxiliary tree is $B_1, .., B_i..., B_n$, the recursive spine is $B_1, .., B_i, .., B_1^*$. E-trees can be combined using two operations: *attachment*

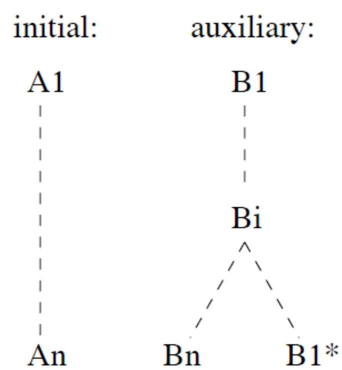


Figure 2.1: Spinal elementary trees

(att) and *adjunction* (adj). Substitution is used to attach an initial tree into a substitution slot of a host tree. Substitution slots are specially marked leaf nodes whose label must match the root of the initial tree. Adjunction is used to attach an auxiliary tree to a node n of a host tree where n must carry the same label as the root and the foot nodes of an auxiliary tree. The two operations are applied to LTAG-spinal e-tree pairs resulting in an LTAG derivation tree (Figure 2.2 taken from (Liu and Sarkar, 2009)) which is similar to a dependency tree. In Figure 2.2, the e-tree anchored with *continue* is the only auxiliary tree; all the other e-trees are initial trees. The arrow is directed from the parent to a child, with the type of operation labeled on the arc. The operation types are: *att* denotes the attachment operation; *adj* denotes the adjunction operation. LTAG-spinal Treebank

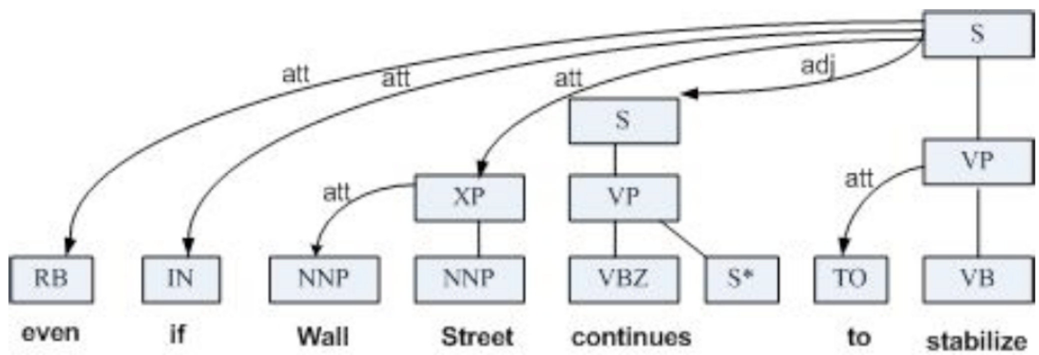


Figure 2.2: An example of LTAG-spinal sub-derivation tree, from LTAG-spinal Treebank

is extracted from the Penn Treebank by exploiting the PropBank annotation. A Penn Treebank syntax tree is taken as an LTAG-spinal derived tree; then, the information from the Penn Treebank and the PropBank is merged using tree transformations. In most cases, the argument is found locally for a predicate in a derivation tree due to the extended domain of locality in e-trees. Figure 2.3 (Figure taken from (Liu and Sarkar, 2009)) shows the 7 most frequent paths from a predicate (P) to an argument (A) in a derivation tree. In

	Path Pattern	Number	Percent
1	P→A	8294	81.3
2	P←A, V←A	720	7.1
3	P←Px→A	437	4.3
4	P←Coord→Px→A	216	2.1
5	P←Ax←Py→A	84	0.82
6	P←Coord←Px→A	40	0.39
7	P←Px←Py→A	13	0.13
total recovered w/ patterns		9804	96.1
total		10206	100.0

Figure 2.3: Distribution of 7 most frequent predicate-argument pair patterns in Sec.22 of LTAG-Spinal treebank. P :predicate, A: argument, $Coord$: coordination and V: modifying verb

Figure 2.2, for predicate *stabilize*, patterns are as follows: *stabilize* → *if*, *stabilize* → *Street*, *stabilize* → *even*, *stabilize* → *continue*, *stabilize* → *to* follow pattern 1. For the predicate *continue*, the pattern *continue* ← *stabilize* follows the pattern 2. Since these 7 patterns account for 95% of the predicate-arguments pairs, for SRL, we consider the candidates which follow one of these 7 patterns to be the candidate arguments of a predicate. For each candidate, features (section. 2.4) are extracted to capture the predicate-argument relations, and a classifier is trained using these features to identify the arguments.

2.3 Architecture

Our SRL system follows a traditional semantic role labeling pipeline of predicate identification, argument identification and classification. In addition, predicate sense prediction and verbose label generation are integrated into the pipeline. Predicate sense prediction and verbose label generation are covered in detail in the section 3.

For argument identification and classification, Support-Vector-Machine (SVM) models are trained on the PropBank annotated Sec.2-21 of the WSJ corpus and features are extracted from the full syntactic parse trees. In the SRL literature, techniques using phrase structure trees, dependency trees and LTAG-spinal trees have been proposed. In this work, we chose LTAG-Spinal trees mainly for two reasons: 1) Extended domain of locality of LTAG-Spinal trees where the arguments of a predicate are locally found 2) Extraction of treebank from the Penn Treebank and the PropBank combined which allowed syntactic and semantic information get embedded in a LTAG-Spinal derivation tree.

We experimented with certain variants of the pipeline in the process of designing the final system. Variants of the pipeline studied are as follows: argument identification followed by classification as a two-stage process, argument identification and classification as the same step, argument classification in conjunction with predicate-sense prediction as a joint model. Performance comparisons of all these variants are reported in the section 2.5.

Tools used

For predicate identification, we have added the predicate identification module of ASSERT (Pradhan et al., 2004) to our pipeline which takes a syntactic parse as input. For feature extraction, SPINC, a bidirectional parser (Shen, 2006), is used for extracting LTAG-spinal tree annotations which takes a sentence and its parts-of-speech (POS) tag sequence as input where the POS tags are extracted using a Stanford POS tagger (Toutanova et al., 2003). For argument identification and classification, we use SVM models trained on the data described in section 2.5.1. There are many implementations of SVM's available such as LibLinear (Fan et al., 2008), LibSVM (Fan, Chen, and Lin, 2005), SVMSGD (Bottou, 2010), Wovpal Wabbit (Langford, Li, and Strehl, 2007) and Megam (Daumé III, 2004). To choose the best of these for SRL, we compared performance accuracies of these SVM implementations on the argument identification task. The performance accuracies are reported in Table 2.1. LibLinear is chosen for its performance accuracy on the argument identification task.

2.4 Feature Selection

The features (Feature definitions taken from (Liu and Sarkar, 2009)) are defined on the predicate-argument pairs from a LTAG-Spinal derivation tree such as predicate e-trees, argument e-trees, intermediate e-trees and topological relationships.

1. **Features from predicate e-tree and its variants** predicate lemma, POS, voice, spine of e-tree and two variants of e-tree by replacing anchor with lemma and voice. In Figure 2.2, for predicate *stabilize* features are: ‘stabilize’ (lemma), ‘VB’ (POS), ‘active’ (voice), ‘S-VP-VB-stabilize’ & ‘S-VP-VB-active’ (variants of spine)

2. **Features from argument e-tree and its variants** argument lemma, POS, Named Entity (NE), spine of e-tree and two variants of e-tree by replacing anchor with lemma and NE, if any. We have found NE a not very useful feature hence we drop this feature. In Figure 2.2, for argument *street*, ‘street’ (lemma), ‘NNP’ (POS), ‘XP-NP-street’ (variants of spine)

3. **Prepositional Phrase (PP) content word of argument e-tree** anchor of the last daughter node if root of the argument e-tree is PP.

4. **Features from the spine node (SP1)** a spine node is the landing site between a predicate e-tree and an argument e-tree. The features include the index along the host spine, label of the node, operation involved (att, adj). The host spine can be an argument e-tree (P←A) or a predicate e-tree (P→A). In Figure 2.2, for predicate *stabilize* and argument *street*, ‘0’(index), ‘S’ (label), ‘att’ (operation).

5. **Relative Position of predicate** before/after

6. **Order** In pattern 1 (P→A), predicate e-tree is the parent and argument e-tree is the child. This feature refers to the order of the argument e-tree among its siblings nodes.

7. **Distance** For pattern (P←Px→A), the distance is 1.

8. **Pattern ID** valued from 1-7 (Figure 2.3).

9. **Combinations** position and pattern ID, distance and pattern ID and position and order.

10. **Features from intermediate predicate e-tree** same features as the predicate e-tree features. In Figure 2.2, for the predicate *continues* and the argument *street*, the path is P←Px→A where the intermediate predicate 'Px' is *stabilize*.

11. **Features from spine node of intermediate predicate e-tree and argument e-tree (SP2)** for predicate-argument pairs of pattern 3-7. These features are similar to the SP1 features but instead between the intermediate predicate e-tree and the argument e-tree

12. **Relative position** between the predicate e-tree and the intermediate e-tree.

13. **Combinations** relative position of argument e-tree and intermediate predicate e-tree and relative position of argument e-tree and predicate e-tree.

14. **Context Features** Lexical and syntactic features in a defined window of a predicate

15. **In Between Features** POS tags of words in between a predicate and argument

16. **Phrase Structure Path** Landing label of nodes on the trace from a predicate to an argument on a LTAG-Spinal tree along with the direction.

2.5 Experiments

2.5.1 Data preparation

We create the training data as follows: Given a LTAG-Spinal parse tree of a sentence with its predicates identified , every node that can be reached following one of the 7 valid paths (Table 2.3) from the predicate is an argument candidate. Hence every such candidate with the features, as described above, and the class label defined by its argument type in Sec.2-21 of the WSJ corpus is a training instance. For the argument classification task, the PropBank argument set labels, A0 to A5 and 13 adjunct-type labels, are used as the class labels. For

Tool	**p/r/f**
LibLinear	95.93/96.85/96.39
LibSVM	77.95/97.21/86.52
SVMSGD	95.56/98.16/96.84
MegaM	96.02/97.40/96.71

Table 2.1: Comparison of SVM implementations

testing, the reference data is generated in a similar fashion from Sec.24 of the WSJ corpus. Due to unavailability of gold LTAG-Spinal trees annotated with semantic roles for Sec.23 of the WSJ corpus, we report scores only on Sec.24.

2.5.2 Evaluation

In order to analyze the impact of automatic syntactic parsers, we compare the performance accuracies using manually annotated trees, also called as gold-standard trees or gold trees, and automatic parsers, also called SPINC trees. The two standard evaluation techniques widely followed are: root/head word based scoring and boundary-based scoring. In root/head word based scoring, a case is counted as positive as long as the root of the argument e-tree is correctly identified. Whereas boundary-based scoring is more strict in that string span of the argument must be correctly identified. We use boundary based scoring for all our experiments and we use CONLL-2004 evaluation scripts for this purpose.

The standard measures for the performance of SRL systems are Precision, Recall and F-score. For each of the semantic role, such as A0, the Precision, Recall and F-score are calculated and overall system performance is evaluated in terms of the number of correctly labeled arguments, the number of labeled arguments and the number of gold arguments. The precision, recall and f-score are calculated as follows:

$$Precision = \frac{number\ of\ correctly\ labeled\ arguments}{number\ of\ labeled\ arguments} \tag{2.1}$$

$$Recall = \frac{number\ of\ correctly\ labeled\ arguments}{number\ of\ gold\ arguments} \tag{2.2}$$

$$F-score = \frac{2 * precision * recall}{precision\ +\ recall} \tag{2.3}$$

Tool	**p/r/f**
Gold Trees	
ident and class	83.47/82.33/82.89
	A0: 91.10/92.97/92.03
	A1: 86.17/88.61/87.37
class	83.23/83.29/83.26
	A0: 90.77/93.55/92.14
	A1: 86.02/89.36/87.66
SPINC Trees	
ident and class	53.63/39.46/45.47
	A0 53.29/42.62/47.37
	A1 59.45/45.47/51.53
class	52.79/40.58/45.89
	A0 52.34/43.39/47.45
	A1 58.69/47.16/52.30

Table 2.2: SRL on gold and automatic parses

2.5.3 Results and Conclusion

All our experiments are conducted on a local machine with Intel dual core processor with 8 GB main memory, training our models on Sec.2-21 of the WSJ corpus takes about 3 minutes and testing on Sec.24 takes less than 5 seconds.

Table 2.1 shows the performance comparison of SVM implementations on the argument identification task using gold-standard trees. LibLinear, SVMSGD and Megam perform equally well on this task. Among these implementations, LibLinear is given preference for its speed, implementation and compatibility with rest of the tool.

Table 2.2 shows the results of the two-stage (*ident and class*) and the one-stage (*class*) SRL using gold-standard trees and automatic parses. With a small feature set, the LTAG-spinal based SRL system described in this chapter provides high precision on gold-standard trees. Experiments using automatic parses shows that the performance is more severely degraded by the syntactic parser. Even though the left-to-right statistical parser that was trained and evaluated on the LTAG-spinal Treebank achieves a f-score of 89.3% on test set, only 81.6% predicate-argument pairs can be recovered from the automatic parses which accounts for the poor performance.

To summarize the chapter: we have introduced the task of Semantic Role labeling and

LTAG-Spinal formalism. We have described our implementation of SRL tool focusing on the engineering aspects of the tool followed by performance evaluations using gold and automatic parses.

Chapter 3

Verbose label prediction

Our main goal in this chapter is to promote the widespread use of semantic parsing of natural language. As non-experts in computational linguistics approach techniques such as semantic role labeling they have to immerse themselves in the jargon of the field, and understand what Arg0 might mean. In this chapter, we aim to automatically produce verbose and easy to understand descriptions in natural language for semantic roles that vary according to the predicate and argument. The information to train such a system already exists in the PropBank Frame Scheme which details type-level information about different predicate senses and their arguments. Using this data, we undertake the task of *Verbose Label Prediction*, which (as far as we know) has not been a direct subject of a detailed experimental study before (although some SRL systems (Koomen et al., 2005) pick a default verbose label in their web-based SRL tool).

In this chapter we define different solutions to the verbose label prediction problem, and show that the task can be done with high accuracy when given accurate SRL information. We test performance of our solutions using the UIUC SRL tool and our SRL implementation. Our approaches improve significantly over strong baselines, demonstrating their viability to verbose label prediction.

3.1 Creation of Training Data

Apart from the sentence level annotations of SRL in the PropBank, there is a little used (by SRL systems) source of information about each predicate in the PropBank. For instance predicate 'accept' in the Penn Treebank has a frame 'accept.01' defined in the PropBank, where ‘01’ is a shorthand for the predicate’s sense, with the following semantic roles and verbose definitions– Arg0:*Acceptor* , Arg1:*Thing Accepted*, Arg2:*Accepted-from* , Arg3:*Attribute*. A predicate can have more than one sense and each such sense has its own set of semantic roles. For every predicate in the Penn Treebank with a different sense, a new frame is created in the PropBank along with verbose descriptions for semantic roles. Figure 3.1 shows two senses of the predicate *kick* from the PropBank. We extract verbose labels from this frame dataset.

	kick.01
"A0":	"kicker"
"A1":	"thing kicked"
"A2":	"instrument"

(a)

	kick.03
"A0":	"contributor"
"A1":	"contribution"
"A2":	"given to"

(b)

Figure 3.1: Verb frames of *kick* from PropBank

We create the training data as follows: Each predicate token in the PropBank is assigned a sense identifier that allows us to match the argument of that predicate to a detailed natural language description about that argument stored in the frames directory of PropBank for the predicate in question. The natural language descriptions are supposed to be human readable, not necessarily machine readable. As a result they can be quite long winded, and so taking the entire description would be too verbose even for our verbose labels, e.g. Arg1 for predicate ‘motivate’ is *decision or attitude being shown to be right* and Arg4 is *benefactive, justified to*. Instead of using such a long label we tokenize the description into smaller chunks, and then pick the most frequent chunk for each argument type and predicate. In the above example, we would pick *benefactive* for Arg4 and if the most frequent verbose label is still too long-winded then we take the first word, e.g. *decision* for Arg1. This results in a verbose label for each argument for a given predicate in the PropBank. The training data has 90,819 predicate instances and our dev (Sec. 24) and test (Sec. 23) sets have 3252 and 5273 instances respectively.

3.2 Models and Experiments

We experiment in two settings: 1) prediction of verbose labels based on the reference PropBank trees (gold semantic role arguments and their spans for each predicate); and 2) prediction of verbose labels based on the output of the UIUC SRL tool and our SRL system. When we train a multi-class classifier in the methods described below, we always use an L1-regularized logistic regression multi-class model. We use the LibLinear package (Fan et al., 2008) to train this classifier. On a local machine with Intel dual core processor with 8 GB main memory, training our models on Sec.2-21 of the WSJ corpus with features (described later in this section) takes about 3 minutes and testing on Sec.24 or Sec.23 takes less than 5 seconds. We train all our models on Sec. 2-21 of the dataset described in 3.1 and we test on Sec. 24 and Sec. 23. Due to unavailability of gold LTAG-Spinal trees annotated with semantic roles for Sec. 23, we report the performance of our approaches using LTAG-Spinal trees only on Sec. 24.

3.2.1 Verbose Label Prediction on Reference SRL

Baseline Methods

We use the following heuristic baseline methods to compare against our machine learning methods for verbose label prediction.
Baseline-1: For an argument, say Arg0, assign the most frequent verbose label across the whole PropBank where frequency is defined as the number of occurrences in the PropBank as a whole. This baseline exploits the fact that verbose labels can remain same even if predicate sense varies.
Baseline-2: For an argument, say Arg0, assign the most frequent verbose label among all the verbose labels for that argument in the list of predicate frames. This baseline pays attention to the predicate when choosing the verbose label.
Baseline-3: Assign the first sense '01' for each predicate and return the verbose label for that argument in this frame. This technique is currently used in the UIUC SRL tool. Table 3.1 shows that '01' is the most frequently used sense for a predicate.
Baseline-4: A predicate frame in the PropBank is a list of arguments for a predicate. We take the list of arguments from the SRL output for each predicate and find the longest match for this list with the frame for each sense of this predicate. For each argument of the predicate in the SRL output, we return the verbose label found in this particular frame. We

break ties by picking the predicate sense that has a lower integer identifier.

Sense	**Instances**
01	2671
02	355
03	117
04	44
05,06,12	11,12,14
07,21	2
08,11	6
09,10,13,14,15,16	1

Table 3.1: Verb Sense distribution of Section 24 of Penn Treebank

Verbose label prediction via sense prediction

One way of solving the verbose label prediction problem is by reducing it to predicate sense prediction. The predicate sense prediction task maps to a multi-class classification task where given a set of senses for a predicate we pick one right sense which mainly depends on its context. The context information to predict a predicate sense can be modeled as features defined on syntactic annotations of the sentence and a model with predicate and its context information as input would be able to generate its sense as an output. We experimented sense prediction approach using features defined on two different syntactic annotations, namely Phrase Structure Trees (PST) and LTAG-Spinal trees. Sense prediction using PST defines a natural extension of the UIUC SRL tool since it uses models trained on PST. Where as sense prediction using LTAG-Spinal extends our SRL system.

Phrase Structure Trees

The context information to predict a predicate sense can be modeled as the features defined in Table 3.2.

In addition to these features, we extend an approach of converting sentence centered at a predicate to canonical form defined in (Vickrey and Koller, 2008) to predict its sense. A canonical form is a representation of a verb and its arguments that is abstracted away from the syntax of the input sentence. For example, “A car hit Bob” and “Bob was hit by a car” have the same canonical form, Verb = hit, Deep Subject = a car, Deep Object = a car. (Vickrey and Koller, 2008) formulated a set of hand-coded transformation rules to convert

⋆ Predicate Lemma
⋆ Predicate Root Form
⋆ Predicate Voice
⋆ Number of Senses
⋆ POS tags on left side of predicate
⋆ POS tags on right side of predicate
⋆ Chunk tags on left side of predicate
⋆ Chunk tags on right side of predicate
⋆ Words on left side of predicate
⋆ Words on right side of predicate
⋆ Siblings of Parent VP

Table 3.2: Verb Sense Disambiguation Features for Phrase Structure Trees

a sentence centered at a predicate to its canonical form.

A transformation rule consists of two parts: a tree pattern and a series of transformation operations. It takes a parse tree as input, and outputs a new transformed parse tree. The tree pattern determines whether the rule can be applied to a particular parse and also identifies what part of the parse should be transformed. The transformation operations actually modify the parse. Each operation specifies a simple modification of the parse tree.

Figure 3.2 (figure taken from (Vickrey and Koller, 2008)) shows an example of depassivizing a sentence centered at predicate *give*. The first part of the rule (represented as a tree) is a tree pattern specifying the constraints on each node and the second part is a set of transformation operations. A transformation operation is a simple step that is applied to the nodes matched by the tree pattern. For example, the "replace 3 with 4" transformation operation applied removes "VP-3" and replaces it with "VP-4". The transformation steps are applied sequentially from top to bottom. Any nodes not matched are unaffected by the transformation; they remain where they are relative to their parents. For example, 'chance' is not matched by the rule and thus remains as child of the VP headed by 'give'. Altogether, there are 154 (mostly unlexicalized) rules. Table 3.3 (Table taken from (Vickrey and Koller, 2008)) shows a summary of the rule-set grouped by type.

The algorithm for canonical form generation of a syntactic parse tree P is as follows: let S be a set of derived parses initialized to P. Let R be the set of rules in Table 3.3. One iteration of the algorithm consists of applying every possible matching rule $r \in R$ to every parse in S, and adding all resulting parses back to S. Rule matching is done top-down; find

Rule Category	#	Original	Simplified
Sentence normalization	24	Thursday, I slept.	I slept Thursday.
Sentence extraction	4	I said he slept.	He slept.
Passive	5	I was hit by a car.	A car hit me.
Misc Collapsing/Rewriting	20	John, a lawyer, ..	John is a lawyer
Conjunctions	8	I ate and slept.	I ate.
Verb Collapsing/Rewriting	14	I must eat.	I eat.
Verb Raising/Control (basic)	17	I want to eat.	I eat.
Verb RC (ADJP/ADVP)	6	I am likely to eat.	I eat.
Verb RC (Noun)	7	I have a chance to eat.	I eat.
Modified nouns	5	Float (The food) I ate.	I ate the food.
Floating nodes	5	Float (The food) I ate.	I ate the food.
Inverted sentences	7	Will I eat?	I will eat.
Possessive	7	John's chance to eat..	John has a chance to eat.
Verb acting as PP/NP	7	Including tax, the total...	The total includes tax.
"Make" rewrites	8	Salt makes food tasty.	Food is tasty.

Table 3.3: Rule Categories with sample simplifications

node that matches the constraints on the root of the tree pattern, then match the children of the root and then their children, etc. The rule set is carefully designed such that no new parses are added with repeated iterations. This simplification is done irrespective of verb hence this process needs to be done only once per sentence.

Naive implementation of this algorithm would result in an exponential number of transformed parses and each such transformation iteration would require copying the whole parse. To alleviate these issues, we make use of an AND-OR tree for storing all transformed parses (S) as defined in (Vickrey and Koller, 2008). It is a general data structure also used to store parse forests such as those produced by a chart parser. Figure 3.4 shows S when we add the depassivized parse in Figure 3.2 back to S which was initialized to the original parse. We can extract all of the possible parse trees from this data structure recursively as follows: each time an OR node is reached, recurse on exactly one of its children; each time an AND node is reached , recurse on all of its children. This way from Figure 3.4 we get original and depassivized sentences. Using this data structure complexity of canonical form generation is reduced to $O(kn^2)$ where k is the number of OR nodes in the forest and n is the number of words in the sentence.

A parse is said to be in a canonical form if it matches with one of the templates shown

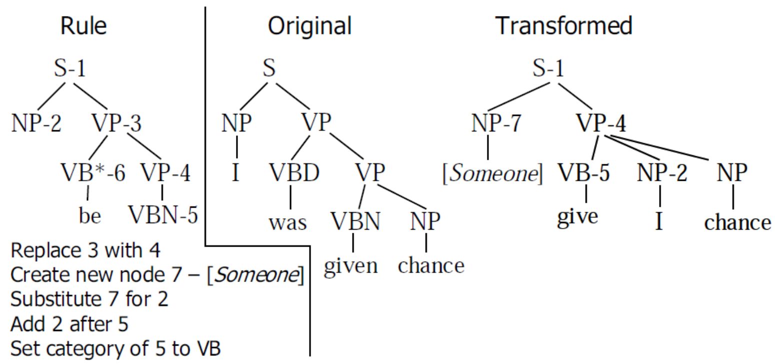


Figure 3.2: Rule for Depassivizing a sentence

in Figure 3.5. These templates define the structure of a simple sentence that have all non-subject modifiers moved to the predicate with predicate as the main verb thereby defining its sub-categorization frame. For example, Figure 3.3 shows a canonical transformation centered on predicate *made* in the sentence “I did not like the decision that was made”. The canonical form “ Somebody made the decision” gets generated with the help of three rule transformations. We can see that sense of the predicate *made* is well defined in the canonical form than in the original sentence. The rule transformations are capturing the structural information, such as position of arguments in the tree, presence/absence of arguments such as *Arg0* for *made* in Figure 3.3, which are useful for predicting the sense.

Hence we use the rules used for canonical form generation as additional features for sense prediction. Table 3.4 shows the performance accuracies of sense prediction task. The baseline for this sense prediction task is to simply pick the default sense ’01’ for each predicate that is defined in current version of the UIUC SRL tool. *VSD Standard* is sense prediction using standard features (Table 3.2) and *VSD Transform* uses the matched hand-coded rules as additional features. We experiment our sense prediction using gold-standard trees and automatic parses generated by Charniak and Johnson parser (Charniak and Johnson, 2005).

LTAG-Spinal Trees

The context information to predict a predicate sense can be modeled as the features defined in Table 3.5.

We add additional features like *Argument Path to Predicate* since the path defined in

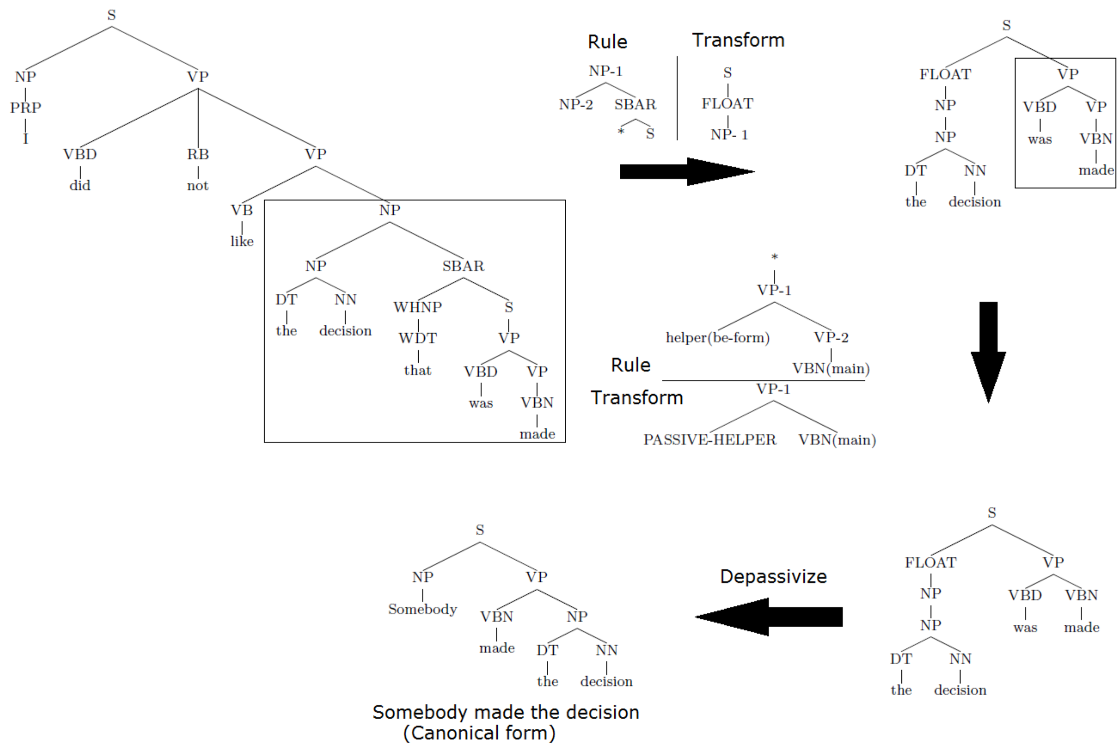


Figure 3.3: Sentence simplification example

LTAG-Spinal trees is less sparse than in PST. Predicate sense can be predicted as a post-processing step of SRL taking arguments identified as features. Alternatively, it can also be done during argument identification/classification process. We perform sense prediction at two different stages in the SRL pipeline, as a post-processing and in conjunction with SRL. Table 3.6 shows the performance accuracies of both these approaches. *VSD Std* experiments refer to sense prediction as post-processing step where as *VSD Joint* refers to joint prediction of arguments of predicate and its sense.

3.2.2 Verbose Label Prediction on SRL System Output

UIUC SRL System

The UIUC SRL tool, apart from being one of the state-of-the-art systems, is one of the few systems to output verbose labels (picked from default predicate frame) for semantic roles. We aim to improve its verbose labeling accuracy. Hence it is chosen as one of

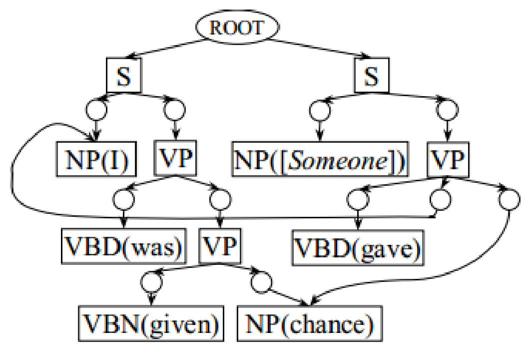


Figure 3.4: Data structure after applying depassivize rule. Circular nodes are OR-nodes and rectangular nodes are AND-nodes

Figure 3.5: Simple sentence template for predicate *eat*

the baseline, defined as *UIUC Baseline-3* in Table 3.7. We obtain SRL output without verbose labels from this system and perform verbose label prediction as a post-processing step generating verbose labels for arguments identified via sense prediction of predicate. To evaluate performance of verbose labeling alone, the accuracy is computed over correctly recognized arguments. We train two models for predicting the predicate sense, one uses only the contextual features as defined in Table 3.2 and another one generates a canonical form of the sentence centered at a predicate and adds it as an additional feature. Both of the models take a predicate and automatically generated syntactic annotation of the sentence using the Charniak parser as input. Table 3.7 shows the performance evaluation of our two models. *Standard* only uses the contextual features where as *Transform* uses the canonical form feature as an additional feature. *Baseline-1-4* are the performance accuracies of the baseline models defined in the Section. 3.2.1.

Approach	**Section-24**	**Section-23**
Default Sense	82.8	82.3
VSD on Gold Parses		
VSD Standard	90.9	90.74
VSD Transform	91.3	90.85
VSD on Automatic Parses		
VSD Standard	90.3	90.1
VSD Transform	90.5	90.3

Table 3.4: Predicate Sense Prediction using PST on Sec. 23 & Sec. 24 of PropBank

⋆ Predicate Lemma
⋆ Predicate Voice
⋆ Number of Senses
⋆ POS tags on left side of predicate
⋆ POS tags on right side of predicate
⋆ Words on left side of predicate
⋆ Words on right side of predicate
⋆ Argument Relative Position
⋆ Argument Head
⋆ Argument Label
⋆ Argument Path to Predicate

Table 3.5: Verb Sense Disambiguation Features for LTAG-Spinal trees

Our SRL System

We obtain the SRL system output from our SRL implementation and used the following two methods for verbose label prediction.

Sense Prediction Given a raw sentence, verbose label prediction proceeds as follows: 1) predicates are identified 2) for each predicate argument candidates are classified 3) sense of a predicate is predicted 4) finally verbose labels are assigned to arguments. Now that we have a set of arguments identified we follow the sense prediction approach for LTAG-Spinal described in Section. 3.2.1 and predict the verbose labels for each argument.

Joint Prediction A joint prediction model is learned for argument classification and sense prediction. For each predicate and for each sense of the predicate we identify argument candidates associated with a classification confidence score and choose the predicate sense

Approach	Section-24
Default Sense	82.8
VSD on Gold Parses	
VSD Std Gold Args	90.58
VSD Std Auto Args	90.55
VSD Joint	71.4
VSD on Automatic Parses	
VSD Std Auto Args	88
VSD Joint	68.5

Table 3.6: Predicate Sense Prediction using LTAG-Spinal on Sec. 24 of PropBank

with the highest confidence score which is the summation of confidence scores of its argument candidates. In this setting, the verbose label prediction is tightly integrated with the SRL system.

Table 3.8 shows the performance accuracies of the two models compared against the baselines defined in the Section. 3.2.1.

3.3 Results

3.3.1 Predicate Sense Prediction

Evaluation measure for predicate sense prediction task is simply the total number of times a correct sense is predicted by total number of predicates. Table 3.4 summarizes the performance of sense prediction using PST. The model which uses canonical form transformation rules (*VSD Transform*) as an additional feature performed marginally better than the standard features (*VSD Std*) in both settings (gold-standard trees and automatic parses) significantly outperforming the baseline of choosing default sense. This shows that the canonical form transformation rules are successful in capturing syntactic structure information useful for predicting the sense.

Table 3.6 summarizes the performance of sense prediction using LTAG-Spinal trees. We experimented with two settings: sense prediction as post-processing (*VSD Std*) and in-conjunction with argument identification/classification (*VSD Joint*). In the first setting, arguments (gold/identified) are provided as additional features to the sense prediction model. Sense prediction as a post-processing step using gold/auto parses performs better

Approach	**Section-24**	**Section-23**
Verbose Label Prediction on Gold Arguments		
Baseline-1	58.2	31.3
Baseline-2	63.5	68.1
Baseline-3	91.01	92.32
Baseline-4	93.34	94.13
Standard	**94.9**	**95.1**
Transform	**95.04**	**95.2**
Verbose Label Prediction on UIUC System Output		
Baseline-1	11.7	11.9
Baseline-2	60.9	57.8
UIUC Baseline-3	91.46	90.51
Baseline-4	93.4	92.6
Standard	**94.7**	**93.9**
Transform	**94.85**	**94**

Table 3.7: Results on the Sec. 23 &Sec. 24 of the PropBank.

than the baseline model but the joint model failed to improve over the baseline.

From Table 3.4 & 3.6, we can conclude that the features (Table 3.2 & 3.5) are able to capture rich syntactic and contextual information useful to predict predicate sense. Sense prediction as a post-processing step to SRL achieves the best performance in both the cases.

3.3.2 Verbose Labeling

Our evaluation measure is simple, how often does an identified argument gets a correct verbose label (there is no need for precision and recall). This type of evaluation is chosen to evaluate the performance of verbose labeling irrespective of the SRL tool performance.

Table 3.7 shows the results of labeling gold arguments and UIUC SRL identified arguments with features derived from gold/auto Phrase Structure trees respectively. *Baselines-1-4* are models as defined in section. 3.2.1. UIUC SRL system which picks default sense is one of the baseline systems (*UIUC Baseline*). The performance of the model using canonical transformation feature (*Transform*) outperforms all other models which is evident from its performance in the predicate sense prediction evaluation. It significantly improves the verbose labeling accuracy of the UIUC SRL tool.

Table 3.8 shows the results of labeling SRL identified by our SRL implementation. "*Sense*

Approach	**Section-24**
Verbose Label Prediction on our SRL System Output	
Baseline-1	3.5
Baseline-2	53.8
Baseline-3	87.6
Baseline-4	87.71
Sense prediction	**90.89**
Joint Model	76.15

Table 3.8: Results on the dev set (Sec. 24 of the PropBank).

prediction" refers to verbose labeling using sense predicted at post-processing stage where as "*Joint*" refers to joint prediction of arguments and senses. Sense prediction as a post-processing performs better than few strong baselines, on the other hand the joint model performs poorly.

To summarize, we have proposed various baseline methods and more informed models to solve this task, and showed that we can identify such verbose labels with 95.2% accuracy if the semantic roles have already been correctly identified.

3.4 Conclusion

Our main goal in this chapter is promote the widespread use of semantic parsing of natural language. As non-experts in computational linguistics approach techniques such as semantic role labeling they have to immerse themselves in the jargon of the field, and figure out what Arg0 might mean. Deep knowledge of syntactic alternations should not be a pre-requisite for the use of natural language parsing tools. The approach we take in this thesis is to provide easily readable verbose labels in natural language so the end users of semantic parsers can see the output and easily recognize the nature of the annotations provided to them. We framed this issue as the task of verbose label prediction, for which we have created a dataset from the PropBank corpus. We have provided several baselines and machine learning methods for verbose label prediction, and showed that we can be highly accurate at 95.2% on data that comes from the PropBank.

Chapter 4

Text Visualization using Verbose Labeling

Text visualization is challenging mostly due to the complex underlying grammatical structure of natural language. Traditionally, highlighting of key terms and their relationships has been a common practise. This approach however completely ignores linguistic structure and makes interactive search difficult. To allow analysts to search for entities related to specific events requires a sophisticated language analysis. For this purpose, state-of-the-art Natural Language Processing (NLP) algorithms can be used. In this chapter, We present a novel browsing interface which uses high-level textual information leveraging linguistic structures extracted by Natural Language Processing algorithms. The goal of this work is to show NLP algorithms can be employed to create better ways of browsing large collections of text documents. To demonstrate this process, we chose human history articles from Wikipedia describing *who did what to whom, when and where*. This *who-did-what-to-whom* structure is defined as a predicate-argument structure. We use our implementation of *Semantic Role Labeling* (SRL) (defined in the Chapter 2& 3) to automatically extract this linguistic information.

The linguistic annotations of text are represented in connected interactive visualizations which is covered in detail later in the chapter. Before going into visualizations we briefly explain why we chose this dataset and how we process the data using SRL to create a light-weight structured back-end database in this chapter.

4.1 Data preparation

As a proof of concept that NLP transformed text improves interactive search through visualization we chose the history domain. History articles describe events which took place some time in the history at a particular place. The event descriptions share a similar linguistic structure of *who did what to whom where and how*. Wikipedia is an open-source repository of such history articles which can readily act as our data source. The history articles indexed on events, such as *Battle of Carthage* (`http://en.wikipedia.org/wiki/Battle_of_Carthage`), are unstructured in the sense that there is no consistency in the structure of such articles nor are the time and place of events explicitly marked. Thus we quickly run into problems like how to automatically extract the articles related only to the history domain and how to automatically identify the time and location of events.

The project "A history of the world in 100 seconds" (`http://www.ragtag.info/2011/feb/2/history-world-100-seconds/`) published a semi-structured dataset by extracting Wikipedia pages indexed on time (`http://http://en.wikipedia.org/wiki/1942`) ranging from 500 BC to 2000 AD along with location information for each event. These pages are a better choice since their structure is consistent, time is explicitly mentioned and events are described in fewer sentences with a link to a source page in Wikipedia with a more elaborate description. We found SRL to be able to generate meaningful structures for these summarized event descriptions since these strictly follow *who-did-what-to-whom* structure. Figure 4.1 is a snapshot of the semi-structured dataset. We call this semi-structured since the event descriptions are flat text for which we assign a structure using Semantic Role Labeling thereby generating a structured database.

location	year	event
37.530233 27.278369	-499	After a failed attack on the rebellious island of ...
37.07583303 22.42083165	-499	Aristagoras seeks help with the revolt against the...
40.156499374 25.83999664	-499	Miltiades the Younger, the ruler of the Thracian C...

Figure 4.1: History in 100 seconds project

4.2 Data Processing

The final step towards creating a structured database is assigning structure to event descriptions. This structure is defined as a predicate-argument structure. To extract this structure we use our implementation of Semantic Role Labeling. Figure 4.2 shows the output of SRL for the input *King Darius III executes Charidemus for criticizing preparations taken for the Battle of Issus . .*

"A0":	"King Darious III",
"A1":	"Charidemus",
"predicate":	"execute",
"A2":	"for criticizing preparations taken for the Battle of Issus",

Figure 4.2: Semantic Role Labeling Output.

SRL tools (Koomen et al., 2005; Pradhan et al., 2004) identify predicates and their arguments and assign role labels such as (*Arg0* and *Arg1*). Our SRL tool, on top of this, automatically transforms roles into text, such as (*Arg0:'killer'* and *Arg1:'entity_killed'*) for this example, that is readable by the user. Since in the history domain *who*, *what* and *Whom* information is more prominent than *How*, we only pick (*Arg0* and *Arg1*) of a predicate. Figure 4.3 shows the final output of SRL combined with time and location information from the semi-structured database. This is one entry in our final structured database. Each sentence might have multiple predicates, hence each predicate and its arguments is an entry in our structured database.

Many event descriptions have more than one sentence with co-reference expressions referring to entities in previous sentences in the same description. Our SRL tool, for such sentences, identifies pronouns as arguments which is not desirable and this happens very frequently in our dataset. To resolve this issue we use a co-reference toolkit (Lee et al., 2011; Raghunathan et al., 2010) and add the reference back to the pronoun expressions.

As for technical aspects of the interface, in order to make the visual interface easy to access with no installation required, we designed the front end as an interactive visual web-based application. The Back end database is stored as json object which can easily be loaded into a javascript program running on a browser.

"A0":	"King Darious III",
"A1":	"Charidemus",
"predicate":	"execute",
"roleA0":	"killer",
"roleA1":	"entity_killed",
"time":	"333 BC",
"latitude":	36.837894,
"longitude":	36.211109,
"title":	"Battle of Issus"

Figure 4.3: The final output from the natural language processing pipeline for the predicate *execute* combined with the temporal identification and geo-location. There is an additional description field which includes the text of the event from Wikipedia which is quoted in the text.

4.3 Visualization

Linguistic annotations along with location and temporal information are plotted onto appropriate interactive visualizations which collectively we call LensingWikipedia. To make the interface easy to access and use with no installation, we designed the interface as a web-application. LensingWikipedia is a novel browser interface which relies heavily on NLP transformed text.

Location information is in terms of latitude and longitude of the event hence we plot on an interactive Map. The time of events ranges from 500 BC to 2000 AD so we provide a Timeline for interactive time range selection. The predicate-argument structure is essentially a tree structure with a predicate as the head and arguments as its children connected using the role types. Hence we plot this information onto an Rgraph (term defined in Javascript InfoVis toolkit `thejit.org`) which is a collection of trees. Our first version of interface has three connected components namely: Map, Timeline and Rgraph (Figure 4.4).

A popular text visualization technique for browsing articles is by organizing articles into hierarchical categories. This classification system enables the articles to be accessed in multiple ways rather than in a single, pre-determined, taxonomic order. This way of navigation using classification is called faceted navigation or faceted browsing where the facets are features or dimensions of the data which defines a hierarchical organization of information.

For our dataset, we create facets using the predicate-argument structures and various other properties. As a replacement for the Rgraph (graph-based text browser), in our second version, we provide a faceted browsing interface along with a map and timeline (Figure 4.6 & 4.7).

4.3.1 Map, Timeline and Rgraph

We collectively visualize location information, temporal information and predicate-argument information about events using three connected visualizations: a geographical view (map), a temporal view (timeline) and a view of the relationship between the predicates and various entities in the text (Rgraph).

Map

Geo-location data allowed us to situate the events on an interactive map (for which we used the Google Maps API) shown in Figure 4.4(a). Every event is associated with a geo-location. Using this information a marker is placed on the map. Adding a marker for every event makes the map slow and less interactive. Instead clustering markers proved to be an efficient solution. The clustering algorithm is simple; for each new marker it sees, it either puts it inside a pre-existing cluster, or it creates a new cluster if the marker doesn't lie within the bounds of any current cluster. The bounds for a cluster (radius of a cluster) is defined based on geo-coordinates of markers as well as map zoom level. Clusters indicate number of events in a region depicted by color and size. Cluster colour scheme is sequential as per colorbrewer (Harrower and Brewer, 2003) depending on number of events in each region. We define four cluster types: 1) Cluster with events greater than 1000 2) Cluster with events greater than 100 and less than 1000 3) Cluster with events greater than 10 and less than 100 4) Cluster with events less than 10.

Maps support interactive browsing by filtering events restricted to a bounding box which is essentially the viewport. The viewport can be changed either by panning or zooming and events in region of interest are displayed. The map can be zoomed either by using a navigation bar provided by Google maps API or by double clicking on a cluster.

Timeline

The temporal identification of each event allowed us to use an interactive timeline shown in Figure 4.4(c). Timeline has a slider for specifying time range, a canvas to situate event

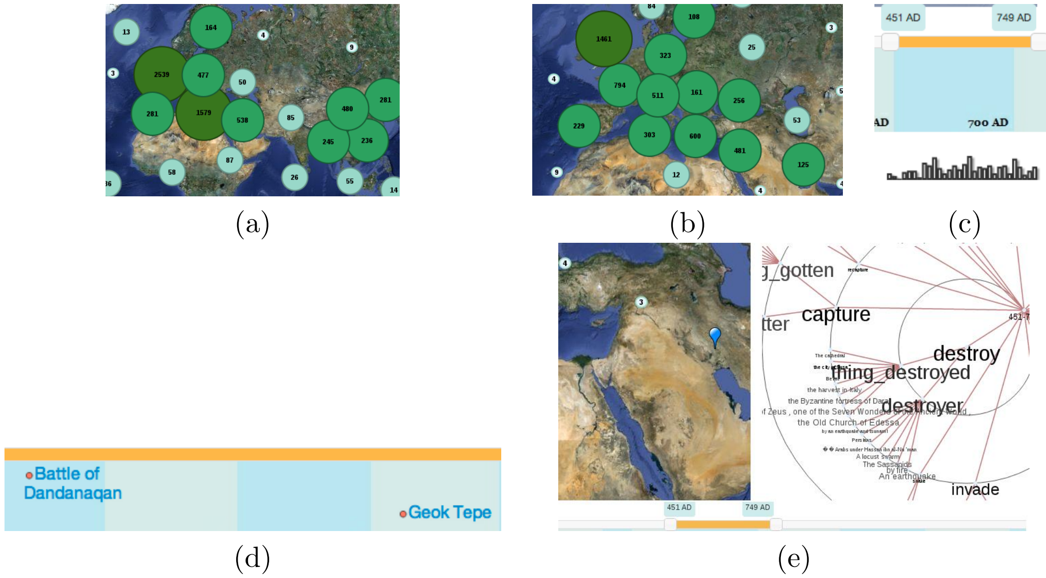


Figure 4.4: Visualizations of events in time and space. Cluster colour scheme in the map view (a)& (b) is sequential as per colorbrewer depending on number of events in each region.

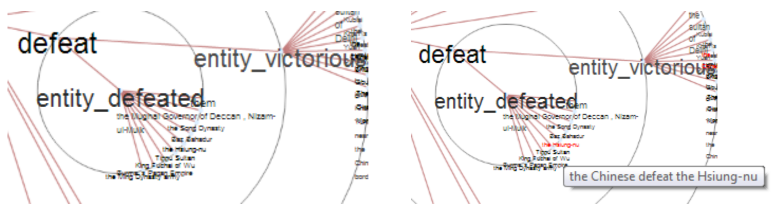
descriptions and a bar graph to show distribution of events (number of events per year) across the time range.

The timeline supports interactive browsing by filtering events restricted to the time range selected on the slider. The timeline canvas displays event descriptions situated chronologically when a cluster on the map with less than 20 events is clicked. These event descriptions are linked to source Wikipedia articles and can be accessed by clicking on them.

Rgraph

Predicate-argument structure is a tree with a predicate as the head and its arguments as its children and a collection of such predicate trees can be collectively visualized as a graph. Hence we plot predicate-argument structures on to a graph-based visualization called an Rgraph. The predicates are linked to their arguments through arguments' verbose role labels. For instance Figure 4.4(e), *destroy* is linked to the verbose label *destroyer* and in turn linked to its arguments. Predicates, arguments and verbose labels are displayed as a tag cloud with different font sizes depending on their frequencies in the dataset.

Rgraph is an interactive browsing interface with predicates, arguments and roles as nodes. Interactive behavior is defined by filtering events based on selection. Selection can be made by clicking either on a predicate (Figure 4.4 (e)), an argument or a role (Figure 4.5 (a)). Focusing on an argument reveals its siblings, in other words predicate-argument structure, with full event description displayed as a tooltip. In Figure 4.5 (b), the sibling of the argument *Hsiung-nu* is shown as a tooltip.



(a) (b)

Figure 4.5: Interactive response of Rgraph

Observations

The three views: map, timeline and Rgraph are all inter-linked to enable a joint exploration of the information in the underlying text data. The natural language descriptions of arguments permits the user to search for intuitive terms such as 'thing_defeated' rather than abstract semantic role labels such as the 'Arg1' of predicate 'defeat' which is what is produced by our SRL system. Using the joint exploration via map, timeline and R-graph, the user can identify individual events of interest which are then shown in the timeline view - Figure 4.4(d). Brushing the event title shows the original sentence(s) from Wikipedia for that event as a tooltip and clicking on the latter takes the user to the Wikipedia page for that event.

Some advantages of browsing using semantic information:

- Easy exploration of events, e.g., to find out how often a specific country was engaged in 'wars', for example, would require selecting events like 'conquer' or 'attack'. And zooming out all the way would reveal all countries engaged in 'wars'.

- To list all 'invaders' of a specific location requires just two clicks, selecting a location and the 'invader' role on SRL view.

- Displaying predicates, arguments and verbose labels as tag clouds in the Rgraph quickly gives an idea of high frequency predicates which gives us a summary of a region history. An interesting statistic is the two most frequent predicates in the data are *found* and *defeat* suggesting our history is dominated by inventions/discoveries and wars.

Some advantages of the interface are:

- Focusing on a location with single click reveals a summary of its history from Wikipedia.
- It is potentially useful for Wikipedia editors to monitor Wikipedia coverage and add missing important events.
- We found that the Rgraph can be used as evaluating the Semantic Role Labeling tool. It can point out some errors committed by the tool in identifying predicates, verbose labels and its arguments.
- We also found that the Rgraph can be used to evaluate the co-reference resolution tool. We found some cases where some pronouns have not been resolved. Error analysis would involve looking at the context of the sentence which is easy using the Rgraph. This can be done by a single click on the argument which displays the full event description.
- Additional information about the distribution of such events across time is provided by the *Bar graph* below the timeline indicating active and passive time frames.

Some of the technical difficulties of this interface affecting browsing are:

- Clustering of markers is done using MarkerClusterer API which redundantly encodes density information as both size and color. MarkerClusterer API calls Google services frequently adding a delay overhead.
- Architecture of this interface is designed such that it loads the full dataset when accessed resulting in a significant loading delay. It also crashes browsers in smart phones since the size exceeds the maximum buffer size allocated to the browser.
- Selecting a time range of 1 year using the slider is difficult since we need to accommodate 2500 integer (500 BC to 2000 AD) selection on a 1300 pixel space giving us a step increase of 2 years per pixel. Timeline is static in the sense that there is no smooth transition response when time range is selected unlike in Google map which has a smooth zooming effect.

- The space in the Rgraph is restricted to only 20 predicates. Exploring the full dataset would require extensive use of the timeline and the map. Also searching for an argument or a predicate or a name is currently not possible.

- Interactive response of the Rgraph is by centering the entity clicked. This is meaningful when a predicate is clicked but not when an argument is clicked. Also centering on arguments pushes its siblings out of the viewport which is not desirable.

To summarize, exploration using semantic information was useful but organizing text as a graph proved ineffective. And browsing suffered due to technical snags. To address the issues mentioned above, we created a new interface with scalable architecture, a new Choropleth style map, a new interactive timeline and most importantly a faceted browsing interface.

4.3.2 Map, Timeline and Facets

The previous version provided interesting insights about the data but suffered mainly due to ineffective way of organizing text in a graph, and due to some technical aspects such as lack of scalable architecture, Choropleth map, Rgraph search and interactive response of timeline.

In this new version, to emphasize the importance of semantic role labeling or in general NLP processing, we employed a text visualization methodology called faceted browsing. We designed a scalable architecture by maintaining a server node which sends data to client only on demand reducing the overhead on the client. The old timeline is replaced with a new timeline with an interactive response. And in addition to the map with clusters, an interactive Choropleth map is provided.

Architecture

To reduce the client overhead, we redesigned the architecture to be a client-server architecture with processing and data storage on the server-side. We maintain a database on the server-side and a client queries the database thereby loading data on demand. As a replacement for the MarkerClusterer API, we implemented server-side stand-alone clustering algorithm which returns only the center and size of cluster reducing memory overhead by a huge margin. This architecture is memory efficient and more scalable than the previous version. We use *MongoDB* (`http://www.mongodb.org/`) and *nodejs* (`http://nodejs.org/`)

for server-side database and client queries handler which is implemented in *javascript* hence a perfect fit for our purpose.

Map

In addition to the map with clusters, we also provide a Choropleth style map (Figure 4.6(a) & 4.6(b)). We color the countries based on number of events with same color scheme as used for clusters. This way we preserve positives of clusters, like the ability to show events on small islands which is not visible in a Choropleth map, and provide additional information. Clickable countries define interactive behavior of this map loading only country specific events. Figure 4.6(b) shows the interactive behavior when clicked on *USA*.

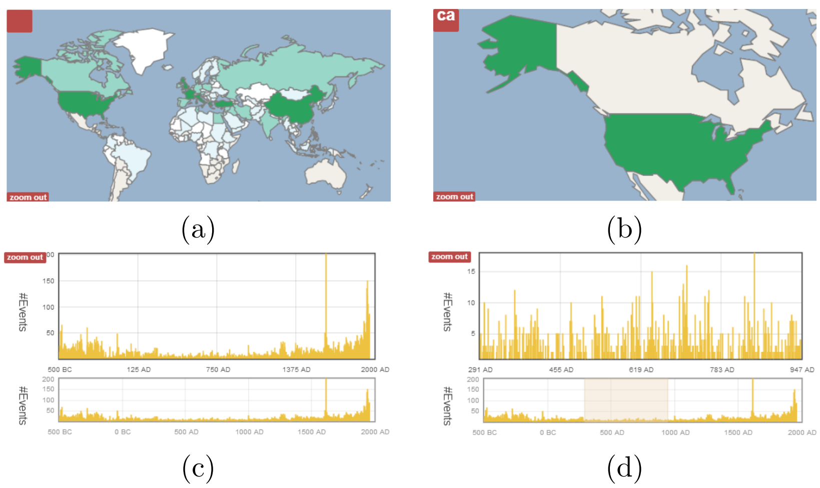


Figure 4.6: Visualizations of events in time and space. Colour scheme to color countries in the Choropleth map view (a)& (b) is sequential as per colorbrewer depending on number of events in each region.

Timeline

Our previous timeline suffered due to lack of interactive response and 1-year range selection. Although the bar graph was very useful. In this version of timeline, we add an interactive response while preserving the utility of the bar graph. A meaningful interactive response on the timeline is to smoothly zoom into the time range selected similar to the zoom effect on Google Maps. We represent the timeline as a sequence of bars with one bar for each year (Figure 4.6(c)). Selecting a time range requires selecting a region of bars and the timeline responds by smoothly zooming into the region of interest (Figure 4.6(d)). For the purpose of panning the timeline, we provide a summary view of entire timeline. In Figure 4.6(b), a time range selection is displayed in summary view below the zoomed-in view. With the help of these two interfaces, there can be no restriction on the number of years that can be shown on the timeline. A one-year time range can be easily selected using the zoomed-in view.

Faceted Browsing

Faceted browsing is a technique for accessing information organized according to a faceted classification system, allowing users to explore a collection of information by applying multiple filters. It has been applied for closed domain datasets like Nobel prize winners, and recipes (Stoica and Hearst, 2007). To our knowledge, this is the first time faceted browsing has been implemented for an open domain dataset like history articles. We classify the event descriptions using predicate-argument structures, named entities and topics generated by automatic topic modeling. In this version, we use the following facets [1]

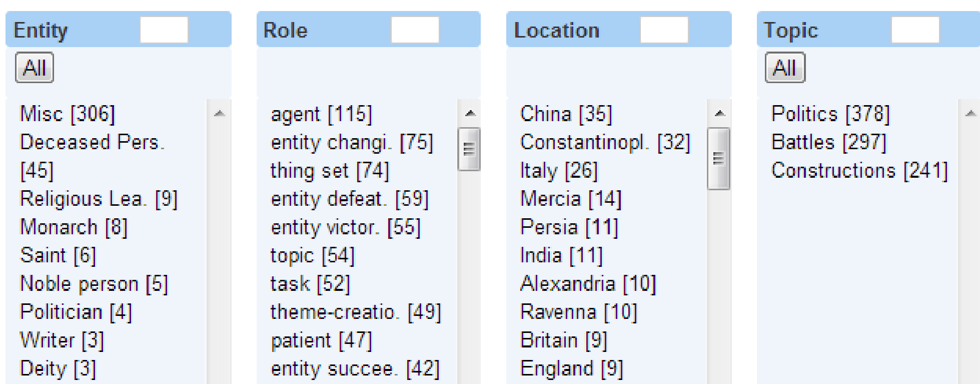
- Named Entities: Presenting events about a person or a location as a story line is an interesting search result and this type of chronological organization of events is a common practise in most Wikipedia articles. To enable such story line presentation, we classify event descriptions based on named entities, such as *person*,*location* and *organization*. It is safe to assume that users are more interested to know what an entity, such as *Alexander*, has done hence we use the entities only if they are part of *Arg0* or *Arg1* of a predicate. Since locations and person names are more prominent, they are put as two different facets.

- Freebase categories: Browsing through a long list of Named entities such as a list of person names is tedious. Hence we add another level of hierarchy by classifying entities using category definitions from Freebase. Freebase is an entity graph of people, places and things created from multiple data sources including Wikipedia. We use category information of entities, for example *Military Commander* and *Author* are some of the Freebase categories for *Adolf Hitler*.

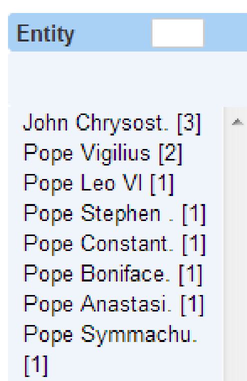
[1] Entity and Topic facets were developed in collaboration with Maryam Siahbani, a Ph.D. in the Natural Language Lab

- Verbose labels: One interesting aspect about the Rgraph is that by clicking verbose label, such as *invader*, we get a list of invaders in history. This shows that a fixed set of verbose labels classifies data in an interesting manner. Hence, we use these labels as one of the facets.

- Predicates: We found that users were quite interested in the Rgraph since it displayed the top few interesting predicates such as *destroy*. Hence the predicates are useful facets but a long list. To create another level of hierarchy on top, we classified events into topics using automatic topic modeling. Topic modeling tool requires the number of topics to classify the data into as input. We experimented with 5, 7, 11 classes and found 5-class classification to be the best. Top few words in each of topic class were mainly predicates. With the help of these we defined three classes namely Politics, Constructions and Wars.



(a)

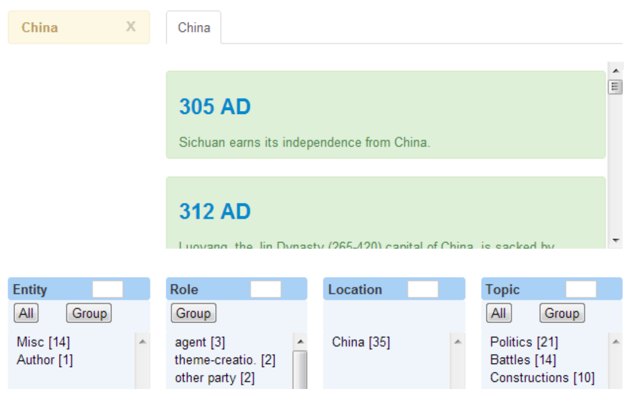


(b)

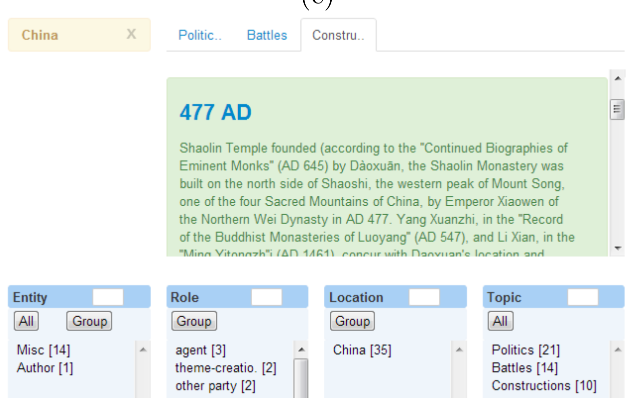
Figure 4.7: Facets

Facets interact with the map and timeline by showing data specific to a selected region. Each entity in the facet is clickable which queries the database and retrieves data related to the clicked entity. In order to support addition and deletion of filters, we explicitly show the selected filters and provide a close button to remove them. We also provide a typeahead search box to quickly search for entities.

Figure 4.7(a) shows all the facets. *Entity* facet has all the person names categorized as *Monarch*, etc. Figure 4.7(b) shows the list of *Religious leaders* category. The *roles* facet has all the verbose labels and the *location* facet has all the location names. The *topic* facet has all the predicates categorized by classes given by topic modeling. Elements in these facets are sorted based on frequencies in the database.



(c)



(d)

Figure 4.8: Faceted browsing interaction

Each element in these facets is a constraint and when clicked a constraint is added and all the documents satisfying this constraint are displayed chronologically. Figure 4.8(c) shows the result of clicking *China*. These constraints can be removed by clicking on the close symbol. We can group the resulting documents based on other facets. In Figure 4.8(d), documents are grouped by topic facet. By default, resulting event descriptions are grouped by selected filters.

Observations

Some advantages of browsing using facets are:

- Faceted browsing provides a more flexible way for exploring the dataset compared to the Rgraph. Constraints can be quickly searched, added or removed in any order. One can quickly narrow down on to a few related documents with fewer constraints.

- The faceted interface simultaneously shows where to go next and how to return to previous states and also provides free text search within the category structure at the same time. It provides an organizing context for results and for subsequent queries which is important for exploration and discovery.

- Topic modeling and Freebase categories provides an interesting level of hierarchy on top of named entities and is a smarter way to browse data.

- Group by is an elegant way of comparing documents. For example, in Figure 4.7(d), *China* is selected as a constraint and results are grouped by *Topic* facet. Here we can quickly compare events where *China* was involved in *Battles* and *Constructions* easily.

Some technical advantages of this version over the previous version are:

- Server-side Processing and loading data on demand optimized memory usage and hence is faster and also works on smart phones.

- Entities of interest can be quickly searched for using the search box

- Choropleth map shows distribution of events specific to geographic boundaries allowing country specific queries.

- Timeline shows an overview of how events are distributed across time as well as in a selected time range. The overview timeline allows smooth comparison of events in two time ranges. The zoomed timeline allows easy selecting of 1-year time range.

Some disadvantages are:

- Verbose label facet has only one-level of hierarchy resulting in a long list of roles to search from. One possible solution is to use the same set of classes that we use for predicates.
- Choropleth map lacks smooth panning and zooming capability. This can be fixed easily by providing a navigation bar similar to the one used in Google Maps.

To summarize, faceted navigation using semantic information and verbose labels made exploration easier . User-intended source articles could be reached easily using fewer filters and NLP information is completely hidden from user. Freebase and topic models provide some interesting categories reducing mental work of searching an information collection. Technically, this version solves most of the problems prevalent in the previous version and is interactive, scalable, efficient and fast.

4.4 Conclusion

Traditionally, text visualizations are restricted to highlighting key words and ignoring syntactic structures. In this chapter, we presented a novel browsing interface which uses high-level textual information leveraging linguistic structures extracted by Natural Language Processing algorithms. The aim of this work was to show that NLP algorithms can be employed to create better ways of browsing large collections of text documents. As a proof of concept, we chose history articles from Wikipedia. As part of data processing, we extracted predicate-argument structure, time and location for each event description. With this data as backend we proposed a novel visualization Map-Timeline-Rgraph (section 4.3.1) where location information is plotted on a map, time information onto a timeline and predicate-argument structure onto an Rgraph. We presented a brief use case study of this interface highlighting advantages and disadvantages. To address the issues of Map-Timeline-Rgraph interface we proposed a Map-Timeline-Facets (section 4.3.2) visualization with scalable architecture and flexible browsing. Our visualizations emphasize the fact that NLP transformed text can improve text visualizations.

Chapter 5

Conclusion and Future Work

This work can be further extended as follows:

- Manual evaluation of verbose labels generated for unseen Wikipedia data can help improve verbose labeling approaches
- Manual evaluation of lensingWikipedia has not been attempted in this thesis but would definitely help understanding effectiveness of verbose labeling in visual browsing
- Thorough error analysis and feature calibrating can give us a better idea in terms of how to increase the performance of our SRL implementation
- It would be interesting to see how well verbose labeling improves text analysis when integrated into text analysis software like CzSaw.

LensingWikipedia can be further improved as follows:

- LensingWikipedia can be extended to accommodate a larger dataset in Wikipedia (`http://en.wikipedia.org/wiki/List_of_years`)
- Updating the database can be automated so that whenever a new article gets added to Wikipedia, the database gets updated.
- Choropleth map can be improved by providing panning and zooming capabilities
- For different applications, the generic set of verbose labels from the PropBank should be manually curated so that domain specific verbose labels are preferred, e.g. the verbose label 'wait_er_not_waiter' could be replaced with 'person_waiting' if the predicate

'wait' is typically applied to animate subjects in the corpus. There are also many 'joke' labels (like the one just mentioned) in the PropBank data which may not translate from linguists to the general populations.

- We currently use Stanford Named Entity Tagger and Freebase for facets but these tools are not very accurate on our data. Instead of Freebase, one can directly utilize the hierarchical categorization called Wikipedia Categories that is part of Wikipedia which is better suited to our task since we are extracting our data from Wikipedia.

In this thesis, we introduced a new task called Verbose label prediction that takes the output of semantic role labeling and associates each of the argument slots for a predicate with a verbose description. We started off by introducing the task of Semantic Role Labeling and Verbose Label Prediction in Chapter 1 and described our implementation of SRL system in Chapter 2 with experimental evaluations. In Chapter 3, we proposed various baseline methods and more informed models to solve the Verbose Label Prediction task and showed that this task can be solved with high accuracy provided correct semantic roles have been identified. We extended our work on semantic parsing to text visualization mainly to illustrate the importance of verbose labeling. As a proof of concept, in Chapter 4, we proposed two interactive browsers for human history articles from Wikipedia, called lensingwikipedia (`http://www.lensingwikipedia.cs.sfu.ca`).

References

Abeille, A. and O. Rambow, editors. 2001. *Tree Adjoining Grammars: Formalisms, Linguistic Analysis and Processing.* Center for the Study of Language and Information.

Bottou, Léon. 2010. Large-scale machine learning with stochastic gradient descent. In *COMPSTAT*.

Boxwell, Stephen A. and Michael White. 2008. Projecting propbank roles onto the CCG-bank. In *LREC*.

Carreras, X. and L. Màrquez. 2004. Introduction to the CoNLL-2004 Shared Task. In *CoNLL*.

Carreras, X. and L. Màrquez. 2005. Introduction to the CoNLL-2005 Shared Task. In *CoNLL*.

Charniak, E. 2000. A maximum-entropy-inspired parser. In *NAACL*.

Charniak, Eugene and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *ACL*.

Collins, Michael. 2003. Head-driven statistical models for natural language parsing. *Computational Linguistics*.

Coyne, Bob and Richard Sproat. 2001. Wordseye: an automatic text-to-scene conversion system. In *SIGGRAPH*.

Daumé III, Hal. 2004. Notes on CG and LM-BFGS optimization of logistic regression.

Fan, Rong-En, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *J. Mach. Learn. Res.*

Fan, Rong-En, Pai-Hsuen Chen, and Chih-Jen Lin. 2005. Working set selection using second order information for training support vector machines. *J. Mach. Learn. Res.*

Fillmore, C.J., C. Wooters, and C.F. Baker. 2001. Building a large lexical databank which provides deep semantics. In *PACLIC15*.

Gildea, D. and M. Palmer. 2002. The necessity of parsing for predicate argument recognition. In *ACL*.

Gildea, Daniel and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*.

Hacioglu, K. 2004. Semantic role labeling using dependency trees. In *COLING*.

Harrower, M. and C. A. Brewer. 2003. Colorbrewer.org: An online tool for selecting colour schemes for maps. *Cartographic Journal*.

Hearst, Marti A. and Emilia Stoica. 2009. NLP support for faceted navigation in scholarly collections. In *NLPIR4DL*.

Koomen, Peter, Vasin Punyakanok, Dan Roth, and Wen tau Yih. 2005. Generalized inference with multiple semantic role labeling systems. In *CoNLL*.

Langford, John, Lihong Li, and Alex Strehl. 2007. Vowpal Wabbit.

Lee, Heeyoung, Yves Peirsman, Angel Chang, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. 2011. Stanford's multi-pass sieve coreference resolution system at the conll-2011 shared task. In *CONLL Shared Task*.

Liu, Yudong and Anoop Sarkar. 2007. Experimental evaluation of LTAG-based features for semantic role labeling. In *In EMNLP-CoNLL*.

Liu, Yudong and Anoop Sarkar. 2009. Exploration of the ltag-spinal formalism and treebank for semantic role labeling. In *GEAF*.

Marcus, Mitchell P., Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: the penn treebank. *Computational Linguistics*.

Melli, G., Y. Wang, Y. Liu, M. Kashani, Z. Shi, B. Gu, A. Sarkar, and F. Popowich. 2005. Description of squash, the sfu question answering summary handler for the duc-2005 summarization task. In *DUC*.

Moschitti, Alessandro, Daniele Pighin, and Roberto Basili. 2006. Semantic role labeling via tree kernel joint inference. In *CONLL*.

Narayanan, S. and S. Harabagiu. 2004. Question answering based on semantic structures. In *COLING*.

Palmer, Martha, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*.

Pradhan, Sameer, Wayne Ward, Kadri Hacioglu, James H. Martin, and Daniel Jurafsky. 2004. Shallow semantic parsing using support vector machines. In *HLT-NAACL*.

Punyakanok, V., D. Roth, and W. Yih. 2005. The necessity of syntactic parsing for semantic role labeling. In *IJCAI*.

Raghunathan, Karthik, Heeyoung Lee, Sudarshan Rangarajan, Nathanael Chambers, Mihai Surdeanu, Dan Jurafsky, and Christopher Manning. 2010. A multi-pass sieve for coreference resolution. In *EMNLP*.

Schabes, Yves and Stuart M. Shieber. 1994. An alternative conception of tree-adjoining derivation. *Computational Linguistics*.

Shen, L. 2006. *Statistical LTAG Parsing*. Ph.D. thesis, University of Pennsylvania.

Shen, L., L. Champollion, and A. Joshi. 2008. Ltag-spinal and the treebank: A new resource for incremental, dependency and semantic parsing.

Shen, L. and A. Joshi. 2008. Ltag dependency parsing with bidirectional incremental construction. In *EMNLP*.

Shen, L. and Aravind Joshi. 2005. Incremental ltag parsing. In *HLT-EMNLP*.

Stasko, John, Carsten Görg, and Zhicheng Liu. 2008. Jigsaw: supporting investigative analysis through interactive visualization. *Information Visualization*.

Stoica, Emilia and Marti A. Hearst. 2007. Automating creation of hierarchical faceted metadata structures. In *NAACL HLT*.

Surdeanu, M., S. Harabagiu, J. Williams, and P. Aarseth. 2003. Using predicate-argument structures for information extraction. In *ACL*.

Surdeanu, M., R. Johansson, A. Meyers, L. Màrquez, and J. Nivre. 2008. The conll 2008 shared task on joint parsing of syntactic and semantic dependencies. In *CoNLL*.

Toutanova, K., A. Haghighi, and C. D. Manning. 2005. Joint learning improves semantic role labeling. In *ACL*.

Toutanova, Kristina, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *NAACL*.

Vickrey, David and Daphne Koller. 2008. Sentence simplification for semantic role labeling. In *ACL*.

Wong, Pak Chung, Beth Hetzler, Christian Posse, Mark Whiting, Susan Havre, Nick Cramer, Anuj Shah, Mudita Singhal, Alan Turner, and Jim Thomas. 2004. In-spire infovis 2004 contest entry. *Information Visualization, IEEE Symposium on*.

Ye, Patrick and Timothy Baldwin. 2006. Verb sense disambiguation using selectional preferences extracted with a state-of-the-art semantic role labeler. In *Australasian Language Technology Workshop*.

Yih, W. and K. Toutanova. 2006. Automatic semantic role labeling (tutorial). In *HLT-NAACL*.