Copyright

by

Vidhoon Viswanathan

2015

The Thesis Committee for Vidhoon Viswanathan

certifies that this is the approved version of the following thesis:

Knowledge Base Population using Stacked Ensembles of Information Extractors

APPROVED BY

SUPERVISING COMMITTEE:

Supervisor: Raymond Mooney

Katrin Erk

Knowledge Base Population using Stacked Ensembles of Information Extractors

by

Vidhoon Viswanathan, B.Tech.

Thesis

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Master Of Science In Computer Science

The University of Texas at Austin

May 2015

To my loving mother Mrs. Uma Viswanathan and father Mr.Viswanathan

Acknowledgments

I am grateful to my adviser Prof. Raymond Mooney for guiding my thesis research. This thesis would not have been possible without his suggestions and inputs. I also thank Prof. Katrin Erk for her valuable feedback on my thesis.

I thank my parents Mr. Viswanathan and Mrs. Uma Viswanathan for their love, support and encouragement to follow my dreams. They have been instrumental in every step of my life.

In addition, I am thankful to several people who helped me in completing my Masters thesis. I wholeheartedly thank UT CS department and faculties for their curriculum, funding opportunities, resources and facilities to support my research work. There was not a single moment where I hit a roadblock and was left unattended. I thank UT Machine Learning Group especially Yinon Bentor, Nick Wilson, Nazneen Fatema Rajani for providing guidance and sharing useful insights related to my research. I also thank David Inouye for teaming up with me for my Natural Language Processing course project which was my first experience working with Prof. Mooney. I appreciate the efforts of Gabor Angeli who provided remote support to get the Stanford Relation Extractor running. I am grateful to Sabarish Kumar Manoharan who inspired me to work on Machine Learning and encouraged me to work on my Masters thesis. I thank Raghavendra Srinivasan and Janaki Kannan for being my well wishers and making Austin feel like home. I also thank my graduate student adviser Prof. Alvisi for his support during my Masters program at

UT Austin. His teaching style and passion for research will continue to inspire me.

Finally, I thank Nishanthi Ravindran for believing in me and encouraging my efforts.

VIDHOON VISWANATHAN

The University of Texas at Austin

May 2015

Knowledge Base Population using Stacked Ensembles of Information Extractors

Vidhoon Viswanathan, M.S.Comp.Sci.
The University of Texas at Austin, 2015

Supervisor: Raymond Mooney

The performance of relation extractors plays a significant role in automatic creation of knowledge bases from web corpus. Using automated systems to create knowledge bases from web is known as Knowledge Base Population. Text Analysis Conference conducts English Slot Filling (ESF) and Slot Filler Validation (SFV) tasks as part of its KBP track to promote research in this area. Slot Filling systems are developed to do relation extraction for specific relation and entity types. Several participating universities have built Slot Filling systems addressing different aspects employing different algorithms and techniques for these tasks.

In this thesis, we investigate the use of ensemble learning to combine the output of existing individual Slot Filling systems. We are the first to employ Stacking, a type of ensemble learning algorithm for the task of ensembling Slot Filling systems for the KBP ESF and SFV tasks. Our approach builds an ensemble classifier that learns to meaningfully combine output from different Slot Filling systems and predict the correctness of extractions. Our experimental evaluation proves that Stacking is useful for ensembling SF systems. We demonstrate new state-of-the-art results for KBP ESF task. Our proposed system achieves an F1 score of 47.

Given the complexity of developing Slot Filling systems from scratch, our promising results indicate that performance on Slot Filling tasks can be increased by ensembling existing systems in shorter timeframe. Our work promotes research and investigation into other methods for ensembling Slot Filling systems.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The World Wide Web overflows with data of all formats. A vast majority of it belongs to text and multimedia formats. Even most of the printed contents published prior to Internet age have been ported successfully to digital age. The sheer volume of this data presents a great challenge in comprehending it. Almost all of this knowledge is rendered useless due to its implicit structural heterogeneity. For the same reason, this knowledge base is termed *unstructured*.

Several efforts are being lead to make data in World Wide Web (WWW) structurally homogeneous. Since it is impossible for humans to manually annotate the entire WWW, there is great interest in automating this organizing task (Surdeanu, 2013) (Surdeanu and Ji, 2014). Specifically, the goal is to build Information Extraction (IE) systems producing annotated data conforming to standard data formats and ontologies defined by humans. This task is referred to as Knowledge Base Population (KBP) (Surdeanu, 2013).

1.1 Relation Extraction

Several extraction systems have been developed to suit different domains and types of information to be extracted. Relation Extraction systems (sometimes known as Relation Extractors) are a particular group of IE systems that focus on extracting relations between entities such as Person (PER) and Organization (ORG).

Inspite of great interest in building Relation Extractors (Bach and Badaskar, 2007), it continues to be a challenging task due to the wide variety of relations possible between different entities. The Text Analysis Conference (TAC) promotes related research in its KBP track and conducts the English Slot Filling (ESF) and Slot Filler Validation (SFV) tasks as part of it. There is great scope for improving the performance of these systems as the state-of-the-art performance is only about 40 (F1 score) (Angeli *et al.*, 2014) in the ESF 2014 task. We use the ESF and SFV tasks in Knowledge Base Population (KBP) track as our benchmark task for all Relation Extraction experiments.

1.2 Motivation for Ensembling

The performance of several systems built to address challenging natural language problems such as parsing (Henderson and Brill, 1998), word sense disambiguation (Pedersen, 2000) and sentiment analysis (Whitehead and Yaeger, 2010) have been dramatically improved by ensembling. In ensembling, multiple systems that take different approaches to solve a problem are combined to improve the individual performance. The performance boost of ensembled system is proportional to the variance in error between different systems (that is, the errors produced by individual systems differ from each other). Hence, if the individual systems have good error variance, the ensembled system is expected to have a significant performance boost (Dietterich, 2000). The KBP ESF and SFV tasks are well suited to investigate ensembling solutions. There are two motivations for this. First, several individual slot filling systems that employ different approaches are already devel-

oped and readily available. Second, slot filling systems are complex and comprise of several components in their pipeline. It requires lot of effort to develop a slot filling system (or relation extractor) from scratch. Hence, exploring methods to ensemble Slot Filling systems might lead to development of better performing systems in shorter timeframe. In this thesis work, we use ensembling to improve the state-of-the-art performance of Slot Filling Systems for KBP English Slot Filling and Slot Filler Validation tasks.

1.3 Contributions of this Thesis

The following are the contributions of this thesis work:

1. We are the first to use Stacking (Wolpert, 1992) for ensembling multiple slot filling systems for KBP English Slot Filling and Slot Filler Validation tasks.
2. We demonstrate new state-of-the-art results from our experimental evaluation. Our ensembled system achieves F1 score of 47 which beats the best KBP ESF 2014 system by 8 points.

1.4 Publications

From the findings of this thesis work, we were able to author the following publication:

1. (Accepted) Vidhoon Viswanathan, Nazneen Fatema Rajani, Raymond Mooney, Stacked Ensembles of Information Extractors for Knowledge Base Population, Submitted to Association of Computational Linguistics 2015, Beijing, China.

1.5 Structure of thesis

The remainder of this thesis document is organized as follows. Chapter 2 presents background needed for remainder of the thesis. Chapter 3 details our proposed approach and Chapter 4 reports our experimental results and discussions. Chapter 5 presents related work. Finally, in Chapter 6 we discuss future work and conclude this thesis report.

Chapter 2

Background

In this section, we introduce the problem of relation extraction. We briefly detail some approaches used to build relation extraction systems. We also introduce and detail two relavant tasks: Slot Filling and Slot Filler Validation, organized under Knowledge Base Population track by the Text Analysis Conference (TAC). Finally, we introduce ensemble learning and provide a brief overview of different methods used for ensembling. All these topics provide necessary background for the remainder of this thesis report.

2.1 Relation Extraction

A relation is defined of the form of a tuple $t = (e_1, e_2, ..., e_n)$ where the e_i are entities in a predefined relation r within a document D (Bach and Badaskar, 2007). Relations could be of any order. Binary relations are common and observed frequently. For instance, *located-in(Austin, Texas)* and *spouse-of(Michelle Obama, Barack Obama)* are examples of binary relations. But higher arity relations are also found in several domains. As an example, *At codons 12, the occurrence of point mutations from G to T were observed* contains a 4-ary biomedical relation *point_mutation(codon, 12, G, T)*.

Relation Extraction can be viewed as a supervised or unsupervised learning problem. Several approaches (Bunescu and Mooney, 2005b) (Culotta and Sorensen,

2004) have been developed to pose relation extraction as a supervised learning problem in which the task is to learn a classifier that predicts if a given relation type is expressed between two entities in a candidate sentence. The major drawback of supervised approaches is the need for labelled training data. The interest in automated systems to populate knowledge bases is to avoid human annotation. But the requirement of human annotation for developing such systems itself is a huge setback. Given the cost and effort involved in producing training data, some approaches (Yan *et al.*, 2009) (de Lacalle and Lapata, 2013) cast relation extraction as an unsupervised learning problem. They apply familiar clustering (Grira *et al.*, 2004) and topic modeling (Blei, 2012) ideas to extract relations. Few other approaches (Yarowsky, 1995) (Blum and Mitchell, 1998) develop semi supervised schemes to extract relations by proposing schemes to generate training data using existing resources.

Relation extraction is a broad problem over different domains with numerous types of relations. This makes it a challenging task and it is equally harder to compare different approaches used to build RE systems. In order to provide an platform to compare different approaches, the Text Analysis Conference (TAC) conducts tasks under its Knowledge Base Population (KBP) track. The goal of these tasks is to promote research of systems that automatically discover information about named entities from unstructured text producing a structured knowledge base (Surdeanu, 2013) (Surdeanu and Ji, 2014). The English Slot Filling (ESF) task and Slot Filler Validation (SFV) task are restricted versions of relation extraction in terms of the number of relations and entity types being considered. Next, we discuss

them in detail as they are the benchmark tasks we use for relation extraction.

2.2 English Slot Filling Task

2.2.1 Task Description

Slot Name	Type	List valued?
per:alternate_names	Name	Yes
per:date_of_birth	Value	
per:age	Value	
per:country_of_birth	Name	
per:stateorprovince_of_birth	Name	
per:city_of_birth	Name	
per:origin	Name	Yes
per:date_of_death	Value	
per:country_of_death	Name	
per:stateorprovince_of_death	Name	
per:city_of_death	Name	
per:cause_of_death	String	
per:countries_of_residence	Name	Yes
per:statesorprovinces_of_residence	Name	Yes
per:cities_of_residence	Name	Yes
per:schools_attended	Name	Yes
per:title	String	Yes
per:employee_or_member_of	Name	Yes
per:religion	String	Yes
per:spouse	Name	Yes
per:children	Name	Yes
per:parents	Name	Yes
per:siblings	Name	Yes
per:other_family	Name	Yes
per:charges	String	Yes

Table 2.1: Slot names, filler entity types and slot type for PER slots

Slot Name	Type	List valued?
org:country of headquarters	Name	
org:stateorprovince_of_headquarters	Name	
org:city_of_headquarters	Name	
org:shareholders	Name	Yes
org:top_members_employees	Name	Yes
org:political_religious_affiliation	Name	Yes
org:number_of_employees_members	Value	
org:alternate_names	Name	Yes
org:founded_by	Name	Yes
org:date_dissolved	Value	
org:website	String	
org:date_founded	Value	
org:members	Name	Yes
org:member_of	Name	Yes
org:subsidiaries	Name	Yes
org:parents	Name	Yes

Table 2.2: Slot names, filler entity types and slot type for ORG slots

The goal of the TAC KBP-ESF task is to collect information (fills) about specific attributes (slots) for a set of entities (queries) from a given corpus. The attributes are similar to relation types in the context of relation extraction. The queries vary for each year of the task and consist of either person (PER) or organization (ORG) entities. The slots are fixed and some are single-valued while others are list-valued i.e., they can take multiple slot fillers. Tables 2.1 & 2.2 list the slots by query and entity types for PER and ORG slots respectively. They also indicates if they are single valued or list valued. There are 25 slot types for PER entities and 16 for ORG entities. In the case of list-valued slots, multiple fillers for the same (query,slot) pair should refer to different entities. It is not just sufficient for them to be distinct strings. For instance, if there are two fillers *Barack Obama* and *Barack*

Hussein Obama for a particular list valued slot of a query entity, then the system must return only one of them since both strings refer to the same person.

2.2.2 Input and Output

Field	Description
id	Query ID
name	String name of entity
docid	Document ID
beg	Start Offset
end	End Offset
enttype	Entity type

Table 2.3: Fields and description for each Query entity

```
<query id="SF13_ENG_001">
  <name>Ramazan Bashardost</name>
  <docid>XIN_ENG_20090916.0146</docid>
  <beg>660</beg>
  <end>677</end>
  <enttype>PER</enttype>
  <nodeid>NIL</nodeid>
</query>
```

Figure 2.1: Sample query for KBP ESF task

The input to the English SF task is set of query entities provided in XML format. Each query consists of the fields listed in Table 2.3. The name and type fields are straightforward. The *document id* field identifies the file in the training corpus where the entity name appears. The *start* and *end offset* fields point to the location in the document where the entity name appears. These fields are useful for disambiguating entities based on the context. A sample query is shown in Figure

Column No.	Description
Column 1	Query Id
Column 2	Slot type
Column 3	Run Id
Column 4	Provenance for relation between query entity and filler
Column 5	Slot Filler
Column 6	Provenance for Slot Filler String
Column 7	Confidence score

Table 2.4: Output fields and description

2.1. In the previous two years (2013 and 2014), there were 100 queries consisting of 50 *PERSON* type entities and 50 *ORGANIZATION* type entities.

For each query entity, the output consists of at least one line for each slot type defined for that entity (from Tables 2.1 & 2.1). In 2014, the task required that each output line must contain seven tab separated columns if it identifies a valid slot filler for that query and slot. Table 2.4 lists each column and the information contained in each column. If no slot filler is identified for a (query, slot) pair, then the output line contains only four fields. The first three columns are straightforward indicating the ID of the query, slot type and unique run ID string for the system. The fourth column contains a NIL string if no valid slot filler was found. If a valid slot filler has been extracted, this field contains a *provenance* for the relation detected. *Provenance* is a triple of the form: $< docid >:< start - offset >:< end - offset >$. The fields are synonymous to their counterparts in query input format. *docid* denotes the document ID that uniquely identifies each document in training corpus. *start-offset* and *end-offset* indicate the location where the relation provenance begins and ends within the document. So, the fourth column contains relation provenance which is the provenance for the relation between the query entity and the

filler entity extracted. Up to 4 provenance triples are allowed and they must be separated by commas. The fifth column contains the filler string. The sixth column contains the provenance for the filler extracted by the system. This follows exactly the same format as the relation provenance. But the number of provenance triples for filler is limited to maximum of 2 entries. While *relation provenance*(Column 4) provides the provenance for the text between the two entities, *filler provenance* (Column 6) provides provenance for the filler text itself. The last (seventh) column contains the confidence score which is a numeric measure between 0 and 1 indicating the confidence of the system in the relation detected between query entity and filler extracted by it.

2.2.3 Scoring

The scoring process is challenging because it is difficult to prepare a comprehensive answer key for all queries and relations from the entire corpus. Hence, this is approximated by pooling the results from all the slot filling systems. Then, human assessors evaluate all the fillers extracted by the system for correctness. In addition, a human generated *manual key* containing answers that are particularly difficult for automated systems to extract is also included in the pooled responses. This is done to promote extraction of such difficult responses. Each non NIL filler extracted is evaluated and belongs to one of the four categories: (1)Correct (2)Inexact (3)Redundant and (4) Wrong.

If the slot filler string contains only part of the answer and includes additional content with the correct answer, then it is marked as Inexact. If multiple slot

filler strings for the same (query,slot) pair represent the same entity, then except one, rest of them are marked as Redundant. If the filler is wrong or if provenance is wrong or if the format is wrong, the filler is marked Wrong. These rules are more elaborately discussed in (Surdeanu, 2013). Using these rules, pooled responses are evaluated and the following counts are computed:

- $Correct$ = Number of correct responses
- $System$ = Number of non NIL responses produced by the system
- $Reference$ = Number of overall correct non NIL responses produced by all systems

Using the above counts, the following performance metrics are computed for official scoring:

- $Precision = Correct/System$
- $Recall = Correct/Reference$
- $F1 = 2 * Precision * Recall/(Precision + Recall)$

The F1 score is the primary metric used for comparing the performance of different systems.

2.3 Slot Filler Validation Task

2.3.1 Task Description

The goal of Slot Filler Validation (SFV) task is to refine the output of individual slot filling systems by either applying more intensive linguistic processing to validate individual candidate slot fillers or by combining the output of multiple slot filling systems. The motivation is to develop a validation component for slot fillers that can be used as part of full Slot Filling system pipeline. The idea is that when slot fillers from multiple systems are available, a validation system could be developed that takes into account the extent of agreement or disagreement between the systems on each output line produced. This could be useful in deciding or modeling the correctness of the slot fillers produced from SF system. Though ensembling (Dietterich, 2000) is not the explicit goal of this task, several participating systems (Wang *et al.*, 2013) (Sammons *et al.*, 2014) use ensemble learning algorithms to achieve a meaningful combination of outputs from multiple slot filling systems.

2.3.2 Input and Output

The input to SFV task is from the English Slot Filling (ESF) task. Each Slot Filling system produces an output file in the format discussed in previous section for every run submitted. The SF output files contain several useful information such as the filler itself, relation and filler provenance, confidence scores for the extraction etc. These SF output files from different systems are grouped together and provided as input for the SFV task. In addition to this, the input queries for

the ESF task consists of an SF systems profile (containing useful details about the systems) and a key file with assessments for a small amount of (query,slot) pairs. To experiment with different configurations of SF systems, each system is allowed upto 5 runs for the KBP ESF task. As a result, the input for SFV task may contain multiple runs from the same system. The SF output files from different systems are anonymized in a way that multiple runs from same SF system can be identified. This is to facilitate system diversity for cross-system voting for an SFV system if needed.

The output of SFV run is a single tab delimited file and follows the same format as Slot Filling system output file. Each line consists of details of a single slot filler extracted by SF systems. In addition to the SF output fields, another field which takes two values: +1 or -1 is added. This field determines if the slot filler is determined as Correct (+1) or Wrong (-1) by the SFV system.

2.3.3 Data

The SFV task also provides previous year KBP SFV task data to facilitate SFV system development. Several SFV systems use this data as training data, for instance, to determine threshold values.

2.3.4 Scoring

Scoring is done in a manner similar to the KBP ESF task. From the output file, we compute the number of correct responses (*Correct*), number of non-NIL responses (*System*) produced by the system and number of correct non NIL responses

produced by all systems (*Reference*). Using them, three performance metrics: *precision*, *recall* and *F1* are computed similar to ESF task. The F1 score is used to compare performance of different SFV systems.

2.4 Ensembling

An ensemble system is a set of classifiers whose individual decisions are combined in some way to classify new examples (Dietterich, 2000). Usually, techniques like weighted or unweighted voting are used to combine the decisions. Generally, ensemble systems end up performing better than the individual classifiers. Hansen and Salamon (1990) identify that the necessary condition for ensemble systems to perform better is that the individual classifiers should be diverse and accurate. An accurate classifier has error rate better than random guessing. Classifiers are diverse if they make errors on different data points.

2.4.1 Ensemble Learning Methods

Several methods are used to construct ensembles. We discuss some prominent techniques here. *Voting* or *Bayesian Voting* is a popular ensemble construction method in which the hypothesis of individual classifiers are weighted by their posterior probabilities to get the final output of the ensemble system. This is useful when the training data is small in which case different hypotheses can be learned for the given problem. Another popular idea is to manipulate training data to develop ensembles. This technique has several specific methods that have been explored. Bagging or Bootstrap aggregation (Breiman, 1996) samples training data with re-

placement into smaller subsets and uses them to construct multiple individual classifiers. The output is then combined using averaging or voting. In ADABOOST (Freund *et al.*, 1996), training examples are weighted. In each iteration, the idea is to minimize the weighted error on the training set and update the weights of training examples accordingly. This results in placing more weights on examples that were misclassified compared to correctly classified examples.

Ensembles can also be constructed by manipulating input features. In this case, individual classifiers are constructed using different subsets of features and their results are aggregated to produce the final output. Finally, we can also modify the output or target classes to create ensemble systems. For this, if there are K output classes, initially the training set is partitioned into two random subsets and labels in one subset are replaced with 0 and other subset is given label 1. The process is repeated for say Ltimes to produce the individual classifiers. Now, using the outcome of all L classifiers, the target class is chosen as the one which received the maximum number of votes.

Chapter 3

System Design and Implementation

In this section, we discuss in detail our Ensemble system for Slot Filling. We do so by first describing the algorithm used for ensembling the output of multiple Slot Filling systems and merging alias extractions. Then, we elaborate on the system design and provide detailed description of various components in the system. Finally, we describe the implementation details of the Ensemble system.

3.1 Ensemble Slot Filling System

Given a set of query entities and fixed set of slots, the goal of an Ensemble Slot Filling system is to effectively combine the output of multiple individual slot filling systems. As input it takes the output of multiple SF systems. This contains several useful information such as confidence scores, filler and relation provenance, source document ID, slot type, query entity type etc. The output of the Ensemble Slot Filling system is a prediction for each input relation extraction indicating if it is Correct (1) or Wrong(0).

In designing an Ensemble slot filling system, there are at least two major tasks. First, we need to design a meaningful way to capture the agreement or disagreement between multiple slot filling systems for different (query,slot) pair extractions. This is important, because, by the statistical principle of ensembling (Dietterich, 2000), this helps in reducing the risk of choosing the wrong slot filling

system and facilitates averaging the output of multiple slot filling systems. Second, we need to identify alias extractions for the same (query,slot) pair. This is important for two reasons. One, by merging such aliass extractions, we combine all available information from multiple SF systems for a (query,slot,entity) triple. Without a merging mechanism, this information could be spread across multiple such triples. Also, the scoring criteria for ESF task marks strings which point to the same entity as redundant and gives no credit. Hence, it is important to merge such extractions. The task of detecting repeated extractions is non-trivial because two different strings could point to the same entity (Example: *Barack Obama* vs *Barack Hussein Obama*).

We achieve these tasks using an ensemble algorithm and an aliasing technique. We discuss them in the following subsections.

3.1.1 Ensemble Algorithm

An ensemble algorithm captures the agreement or disagreement of multiple slot filling systems when provided with relevant features. We choose Stacking (Wolpert, 1992) or *stacked generalization* as our ensemble algorithm to achieve this. Stacking has provided promising performance in several previous ensembling works (Sigletos *et al.*, 2005) (Sill *et al.*, 2009) but it has not been explored to ensemble SF systems. In the context of SF systems, we use Stacking to combine useful information present in the output of multiple slot filling systems.

The basic idea in stacking is to learn a meta-level classifier (or level 1 classifier) based on the output of base level classifiers (level 0 classifiers) (Sigletos *et al.*,

2005). In our ensembling problem, the individual SF systems form the base level classifiers. So, a *stacked meta-classifier* is a binary classifier that takes the output of individual SF systems as input and predicts if the extraction is Correct (1) or Wrong (0). The output of individual SF systems contains several pieces of useful

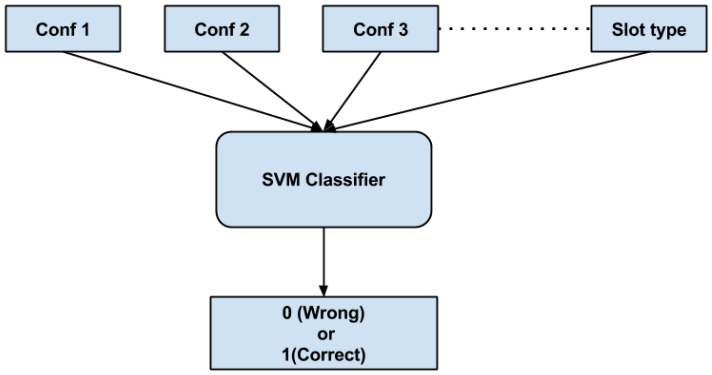

Figure 3.1: Stacked Meta-classifier

information. In this work, we utilize the confidence scores and slot type associated with each extraction from SF systems as input to our stacked meta-classifier. This is meaningful because of the following reasons. First, a high (low) confidence score means according to the corresponding SF system, the relation between (query,slot) pair is more (less) probable. If an extraction has high or low confidence scores from all SF systems, it implies that they agree on the extraction. But if some systems have high confidence scores while others have low confidence scores, then they disagree on the extraction. Hence, a meta classifier taking confidence scores from individual SF systems as input learns to minimize the risk of choosing the wrong SF system output by properly weighting the confidence scores of individual systems for particular slots. This exactly meets our ensembling goal which is to

capture the agreement or disagreement between the individual SF systems. Second, if an extraction is missing from some SF systems, it can be easily represented with a 0.0 confidence score which in turn translates to, *The SF system has no confidence in the relation between the (query,slot) pair*.

Figure 3.1 provides a pictorial view of Stacking applied for ensembling an output extraction from multiple SF systems. Our stacked-meta classifier is a binary classifier that takes confidence scores from individual SF systems and slot type as input and predicts if the extraction is Correct (1) or Wrong (0). We use a single meta-classifier for all slots and entity types. This meta-classifier can be trained using any machine learning model such as Logistic Regression, Naive Bayes or Support Vector Machines (SVM). All the classifier models performed equivalently and we randomly chose an L1-regularized linear kernel SVM (Fan *et al.*, 2008) for our stacked meta-classifier.

3.1.2 Aliasing

When combining the output of multiple SF systems, it is possible that some extractions are aliases of each other. An SF system A could extract a filler $F1$ for a slot S while another SF system B extracts another filler $F2$ for the same slot S. If the extracted fillers $F1$ and $F2$ are aliases (i.e. different names for the same entity), then the two extractions are *alias extractions*. Identifying alias extractions for a (query,slot) pair provides additional information spread across different extractions and also eliminates *redundant* extractions.

To detect alias extractions, we employ a technique derived by inverting the

scheme used by the LSV (Roth *et al.*, 2014a) SF system for query expansion. The LSV SF system uses a Wikipedia anchor-text model (Roth and Klakow, 2010) to generate aliases for given query entities. By including aliases for query names, the SF systems increase the number of candidate sentences fetched for the query.

The Wikipedia anchor-text model (Roth and Klakow, 2010) uses the text associated with anchor tags in Wikipedia to find aliases for an entity. For a given entity, the text and frequency of anchor links in Wikipedia that point to the corresponding Wikipedia article of the entity are computed. This Wikipedia Link data is available readily with the LSV system (Roth *et al.*, 2014a) and can be parsed to obtain the required aliases. Using this data, the top N aliases for the given entity are identified.

To eliminate filler aliases, we apply the same technique and generate aliases for all slot fillers of a given (query,slot) pair. We choose N=10 (top 10 aliases) for our Alias merging system. Given a slot filler, we obtain the corresponding Wikipedia page for the filler entity and retrieve the top 10 anchor texts. They are added to the aliases pool for the (query,slot) pair. The same process is repeated for other fillers for the same (query,slot) pair. Any other filler for the same (query,slot) pair that overlaps or matches with any entry in the aliases pool are marked as alias extractions.

We explored two methods to handle alias extractions. Our first approach was to detect and filter alias extractions from the output of the ensemble meta-classifier. Our second approach was to merge the alias extractions present in the input given to the ensemble meta-classifier. The alias extractions are merged as

follows. The filler string is chosen as the one extracted by the most number of SF systems. The missing confidence scores for this filler string are updated from the confidence values of alias extractions. The output of alias merging is provided as input to the stacked-meta classifier. The performance of these two schemes was statistically indistinguishable and we use the latter approach for our experiments with aliasing.

3.2 System Design and Components

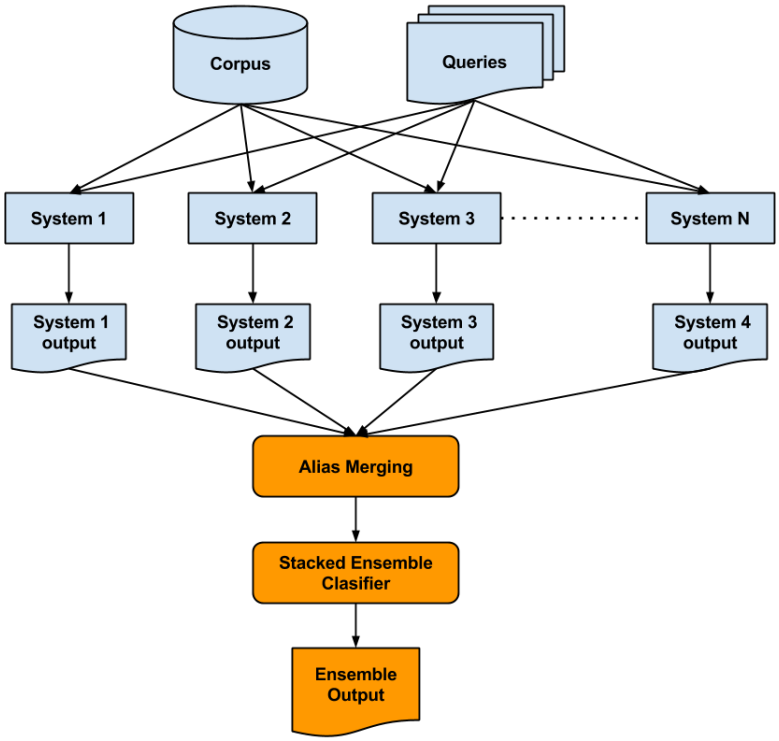

Figure 3.2: Ensemble SF system: Holistic View

Figure 3.2 provides a holistic view of the Ensemble Slot Filling system with

all associated elements. The Alias Merging and Stacked Ensemble Classifier components belong to the Ensemble SF system. It can be seen that individual SF systems are run on the provided SF queries using the supplied corpus to produce their respective outputs. The Ensemble classifier takes merged input and produces Ensemble output which contains predictions classifying the supplied extractions as Correct or Wrong.

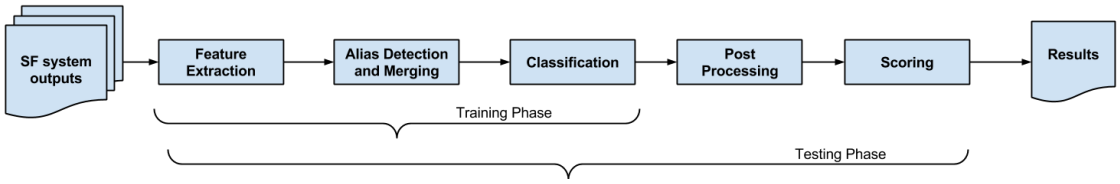

Figure 3.3: Ensemble SF system: Components and Pipeline

Figure 3.3 presents the pipeline showing the various components involved in the operation of the Ensemble SF system. First, the outputs from different SF systems are passed through a Feature Extraction component. This component is responsible for putting together confidence values of each filler for every (query,slot) pair from multiple SF systems. The Feature Extraction component produces a single output file comprising of all unique extractions from multiple SF systems with the confidence values from the respective systems. This component is responsible for inserting 0.0 confidence values for fillers that are missing in the output of some SF systems.

The output of Feature Extractor is passed to the Alias Detecting and Merging component. This component functions as described in previous section and merges all alias filler entries for a given (query,slot) pair. The output of alias merging

component is supplied as input to the stacked-meta classifier.

The stacked-meta classifier outputs a prediction for each given extraction. The post processing component collects all extractions predicted as Correct by the ensemble classifier and puts them in the required SF output format. Finally, the scorer runs the official scorer of Slot Filling task to determine the performance of Ensembled SF system.

3.3 Implementation Details

All components of the Ensemble SF system were implemented in Java. For training and testing the ensemble classifier, we used the Weka (Hall *et al.*, 2009) package version 3.7. Weka provides a wide array of machine learning algorithms implemented and ready to use. We chose the Support Vector Machine (SVM) model for our ensemble classifier. Weka internally uses the LibSVM (Chang and Lin, 2011) implementation for its SVM classifier. We used L1-regularized linear kernel SVM (Fan *et al.*, 2008) as our stacked meta-classifier. For this, in Weka, we set the cost parameter to 1 and chose kernel type as linear. All other parameters use their default values. The scorer is implemented in Java and provided as part of the KBP ESF task.

Chapter 4

Experiments and Results

In this chapter, we describe our methodology for evaluating our ensembled SF system. We begin by explaining the methodology followed and goals for evaluating our system. We continue describing the datasets used and detail the baselines used for comparing the performance of our systems. We also introduce our setup for running experiments. We finally present results from our experiments and discuss our observations.

4.1 Experiments

4.1.1 Methodology and Goals

We run two sets of experiments to evaluate our ensembled slot filling system. In the first set, we use the official key provided by KBP ESF 2014 task. This facilitates comparison with other ESF systems that took part in 2014. In the second set, we use an unofficial key obtained by pooling the results of all KBP ESF 2014 participating systems. There are two motivations for choosing this evaluation. First, it removes results added by humans which are particularly hard for automated systems to extract. Since these extractions are not going to be generated by ensembling the results of individual SF systems, removing these results are going to have the same impact on individual SF systems. Second, previous work on ensembling SF systems (Sammons *et al.*, 2014) uses this key to generate their results. Hence, using

this unofficial key will generate results comparable to their system. In both set of experiments, we compare the performance of system with and without performing Alias merging.

Using these two sets of experiments, we produce results that can be compared with official scores from KBP ESF task and results reported from other comparable ensembled SF systems.

4.1.2 Datasets

LSV	IIRG
UMass_IESL	Stanford
BUPT_PRIS	RPI_BLENDER
CMUML	Compreno
NYU	UWashington

Table 4.1: Common systems which participated in KBP ESF 2013 and 2014

As our ensembled slot filling system consists of a stacked meta-classifier, it must be trained with suitable training data. To evaluate the performance of our ensembled slot filling system, we require both training and test data. Hence, we use the KBP SFV dataset from 2013 and 2014 to identify the common systems that took part in both years. We found 10 common systems and they are listed in Table 4.1. Since, each system has varying number of runs in 2013 and 2014, we choose only one run from each common system and build our common systems dataset ($CDATASET$). We treat these systems as *black boxes* and do not discriminate them based on their approach or relative performance. We use 2013 data for training and 2014 data for testing our ensembled slot filling system. Although systems

change from year to year, our results indicate that the learning transfers with this change.

We also use KBP SFV 2013 and 2014 dataset which consists of outputs from all individual SF systems in the respective years ($FDATASET$). We use this complete dataset for baseline approaches that are not restricted by training process since they will not require compatibility between 2013 and 2014 datasets. The $FDATASET$ consists of 52 outputs from 2013 and 66 outputs from 2014. These outputs also include multiple runs from the same system.

4.1.3 Baselines

We choose two baselines to compare the performance of our ensembled slot filling system. First, we compute a union baseline ($Baseline\_Union$). This is an $all - YES$ scheme in which we include all unique extractions from individual SF systems in the ensemble result. If the slot is single-valued, then we choose to include the extraction with highest confidence. This scheme is expected to produce low precision but high recall results. This baseline is computed from the 2014 $FDATASET$ since the restriction of the training process does not apply to them. Second, we include a voting baseline ($Baseline\_Vote$). For this baseline, we use 2013 $FDATASET$ to identify a threshold on the number of individual SF systems that must agree to include a candidate extraction. Using this learned threshold, we compute the results for the voting baseline on the 2014 $FDATASET$.

4.1.4 Setup

Our experimental setup toggles between Java modules and the Weka Machine learning toolbox. We perform feature extraction and apply alias merging to our datasets using Java implementations. We train our ensemble classifier in Weka using the training portion (2013) of the datasets. After the training process, we evaluate our test data on the trained meta-classifier in Weka and obtain the prediction results. We switch back to our Java modules for post processing the predictions from ensemble classifier and feed it to the scorer to generate the results.

4.2 Results

4.2.1 Results using Official Key

System	P	R	F1
Baseline_Union	0.066	**0.762**	0.122
Baseline_Vote	0.641	0.288	0.397
Best ESF system 2014 (UIUC)	0.585	0.298	0.395

Table 4.2: Baseline results and Best ESF system (2014)

Approach	With Aliasing			Without Aliasing		
	P	R	F1	P	R	F1
Stacking	0.632	0.34	0.441	0.637	0.339	0.442
Stacking + Slot Type	0.642	0.371	**0.47**	**0.646**	0.369	0.469

Table 4.3: Results with Official Key

Table 4.2 and 4.3 present results from our first set of experiments. These are obtained using the official key provided by KBP ESF 2014. The entries in Table 4.2

present the baseline evaluations using just the $FDATASET$ and the performance of the official best scoring system from KBP ESF 2014 task. The entries in Table 4.3 indicate our two approaches: (1) Stacking with only confidence values (2) Stacking with confidence values and slot type. We also present results with and without performing Alias merging. Our approaches use $CDATASET$.

4.2.2 Results using Unofficial Key

System	P	R	F1
Baseline_Union	0.053	**0.872**	0.101
Baseline_Vote	0.637	0.406	0.496
Best SFV system 2014 (UIUC)	0.457	0.507	0.481

Table 4.4: Baseline results and Best SFV system (2014)

Approach	With Aliasing			Without Aliasing		
	P	R	F1	P	R	F1
Stacking	0.632	0.482	0.547	0.637	0.481	0.548
Stacking + Slot Type	0.642	0.526	**0.578**	**0.646**	0.523	0.578

Table 4.5: Results with Unofficial Key

Tables 4.4 and 4.5 present results from our second set of experiments. These are obtained using the unofficial key obtained by pooling all KBP ESF 2014 participating systems. The first two entries in Table 4.4 present the baseline evaluations using just the $FDATASET$. Note that the Union baseline does not produce recall of 1 even after removing human assessments from the key. This is because, we choose the slot fill with highest confidence for single valued slots. The last entry reports performance of the official best scoring system from KBP SFV 2014

task. Table 4.5 presents performance of our two approaches: (1) Stacking with only confidence values (2) Stacking with confidence values and slot type. We also present results with and without performing Alias merging. Our approaches use $CDATASET$.

4.3 Discussion

From the results, it can be observed that stacking is effective as an ensembling method for slot filling systems. The stacked meta-classifier using just the confidence values produces an F1 score of 44 using the official key. This is better than the best KBP ESF system in 2014 (Angeli *et al.*, 2014) by 5 points. In spite of using extractions from all ESF 2014 systems ($FDATASET$), the union baseline achieves F1 score of 12 while the voting baseline achieves F1 score of 39. The performance of our ensembled SF system beats both the baselines at least by 5 points in F1 score.

The slot type feature provides an additional improvement of 3 points in F1 score. The stacked meta-classifier taking slot type along with confidence values achieves an F1 score of 47. This is 8 points greater than the best ESF system in 2014. This indicates that additional features for the stacked meta-classifier could enhance the performance of the ensembled SF system. From the results, it can be observed that alias merging did not have any impact on the performance of the system.

A similar trend is observed from the results using the unofficial key. In this case, our stacked meta-classifier beats the top performing system (UIUC) (Sam-

mons *et al.*, 2014) in the KBP SFV 2014 task. The simple stacking approach with just the confidence values provides an F1 score of 54 which is better than UIUC system by 6 points while stacking with confidence values and slot type achieves an F1 score of 57 while in comparison performs better than UIUC system by 9 points in terms of F1 score.

As expected the union baseline achieves best recall, while our stacking scheme using confidence values and slot type achieves best precision and overall best F1 score.

Chapter 5

Related Work

In this section, we will discuss in detail literature relevant to relation extraction and the problem of ensembling relation extractors. First, we talk about different relation extraction systems and approaches used for developing English Slot Filling systems. Then, we briefly describe how ensembling has been applied to different natural language problems. We also detail different approaches used to combine relation extractors and slot filling systems.

5.1 Relation Extraction

In this section, we review different supervised, unsupervised and semi-supervised methods to perform relation extraction from text data. Most of the methods involve extracting binary relations while there are other techniques that focus on higher order relations. But we will not be covering higher order relations in this report.

5.1.1 Supervised Methods

Given a sentence $S = w1, w2, ..., e1, ...wj, ..., wn$ where $e1$ and $e2$ are entities, the problem of relation extraction for a particular relation R can be posed as a supervised classification task. In that case, the goal is to learn a mapping function $f(.)$ which determines if a given sentence describes a relation between two entities. The mapping function can be represented as:

$fR(T(S)) = 1$, if relation R exists between entities $e1$ and $e2$ in sentence S

0, otherwise

where, $T(S)$ is the set of features extracted from sentence S. Thus, the mapping function detects if relation R is represented in sentence S between entities $e1$ and $e2$. If we have enough positive and negative samples, then any well known classifier like logistic regression, Support vector machines etc. can be trained to perform this classification task. The supervised methods for relation extraction are further divided into two categories based on the input provided to the classifiers namely, Feature based methods and Kernel based methods.

Several syntactic and semantic features can be extracted from a sentence to detect the representation of a relation. A large number of such features have been explored in Kambhatla (2004), Zhao and Grishman (2005) & GuoDong *et al.* (2005). Some of the popular syntactic features used for this task include:

1. the entities
2. entity types
3. word sequence between entities
4. number of words between the entities
5. path in the parse tree containing the entities.

Similarly, semantic features like the dependency path between the two en-

tities in the sentence have also been used. Kambhatla (2004) build a maximum entropy model using some of the features described before for the *Automatic Content Extraction* (ACE) Relation Detection Task, which is an evaluation conducted by NIST. About 24 relation types are used in ACE and only 6 of them are symmetric in nature. While the number of features used in Kambhatla (2004) are limited, Zhao and Grishman (2005) use kernels like argument kernel, bigram kernel and dependency path kernel to include syntactic features. These kernels can be extended to include higher order features. Support Vector Machines (SVM) are used to decide which of these features are important. GuoDong *et al.* (2005) also train an SVM classifier with some of the standard syntactic features but do not use any specialized feature based kernels.

Kernel based methods have also worked well for relation extraction. String kernels proposed in Lodhi *et al.* (2002) have been used extensively in text classification. Given two strings x and y, string kernel computes their similarity based on the number of subsequences that are common to both of them. Using dynamic programming, the complexity for computing the kernel function has been demonstrated to be much less that the expected exponential complexity (Lodhi *et al.*, 2002). The principle of string kernels has been successfully applied to the problem of relation extraction in different works (Bunescu and Mooney, 2005b) (Culotta and Sorensen, 2004) (Zelenko *et al.*, 2003). It is illustrated as follows. For relation extraction, if x^+ and x^- are positive and negative examples for a given relation and y is a test sample, then with kernel K, we can determine if y represents the relation based on the values of $K(x^+, y)$ and $K(x^-, y)$. If $K(x^+, y) > K(x^-, y)$, then y represents

the given relation and if $K(x^+, y) < K(x^-, y)$ otherwise. In the case of relation extraction, the kernel could compute similarity between x+, x- and y in terms of characters or words or parse trees. The kernel based on words is known as *bag of features kernel* and the kernel based on parse trees is called as expected, *Tree kernels*.

Bunescu and Mooney (2005b) use a bag of features kernel based on the idea that context around entities can be used to determine if a given relation exists between them. In particular, they use the word context around the entities. That is, given a sentence $S = w1...e1...w2...e2...w3$, the proposed method utilizes the context before, in the middle and after the entities. They develop three word based kernels to compute the similarity for word-context of the above three types in a manner similar to string kernels. Then, they use another combined kernel that sums the sub kernels to compute the overall similarity and use this to determine relations between entities.

Tree kernels have also been used for specific relations like *person-affiliation* and *organization-location* by Culotta and Sorensen (2004). As described above, in the case of tree kernels, the subsequences are replaced by parse trees to compute similarity between sentences. Culotta and Sorensen (2004) use shallow parse trees to leverage their relative stability and robustness. Zelenko *et al.* (2003) explore the use of dependency parse trees for the same purpose. Overall, tree kernels compute the structural similarity or commonality between sentences which have proven useful in predicting certain relation types. Like subsequences, the similarity measure is proportional to the number of common parse subtrees between the two given parses.

Bunescu and Mooney (2005a) refine this tree kernel approach by determining that dependency path between two entities carries sufficient information to determine the relation between them.

5.1.2 Semi Supervised Methods

In spite of the simplicity of supervised methods, an important shortcoming is the requirement for labelled data to train classifiers. This has been recognized as a critical issue in developing relation extraction systems as produced labelled training data is expensive, labor intensive and error prone. Several systems have been proposed to do overcome this shortcoming by posing relation extraction as a semi supervised or unsupervised problem. Yarowsky (1995), Blum and Mitchell (1998) and Surdeanu *et al.* (2012) propose methods which facilitate training classifiers in a semi supervised way to perform relation extraction. The crux of the approach by Yarowsky (1995) is to train a weak classifier from a limited labelled data and use the output this weak learner to incrementally re train in several iterations and produce a classifier for relation extraction with a desirable performance. Blum and Mitchell (1998) use Co-training where limited labelled training data is used to produce two different conditionally independent views (ie., disjoint feature sets). Classifiers are individually trained on each view and most confident predictions are used to label unlabelled data. In this manner, more labelled data is produced to train the final classifier for relation extraction. Surdeanu *et al.* (2012) propose distant supervision in which the focus is to use open databases such as DbPedia (Lehmann *et al.*, 2014) and Freebase (Bollacker *et al.*, 2007) to produce labelled data for training a relation

extractor.

Dual Iterative Pattern Relation Expansion (DIPRE) (Brin, 1999) and Snowball (Agichtein and Gravano, 2000) are two somewhat similar semi supervised relation extraction systems. The former focuses on $(author, book)$ relations while the latter identifies $(organization, location)$ relation. Both systems start with a small set of seed relations and build a pattern matching classifier. Using the seed relations, they crawl the web to extract more sentences containing those entities and generate a 5-tuple record for each sentence: $< order, author, book, prefix, suffix, middle >$ or $[prefix, organization, middle, location, suffix]$. In DIPRE, the extracted tuples are grouped based on some fields in the tuple such as middle and patterns are induced by taking the longest common prefix and suffix strings. The newly induced patterns are used to crawl more sentences for the the next iteration of the process. This process is repeated till no new sentences are extracted by the induced pattern. Snowball follows a similar process except that it employs a similarity function instead of exact matching leading to more flexibility in the matching mechanism. Snowball also employs a Named Entity Recognition (NER) system to identify all location and organization entities in the data and then uses the sentences with location, organization entity pairs to induce new patterns for extraction the relation.

Both DIPRE and Snowball focus on specific relation types. In order to build a generic relation extraction system, Banko *et al.* (2007) proposed TextRunner which does not require the relation and its format to be specified as input. It consists of three components which work in a pipeline namely learner, extractor and assessor. The learner uses a 7-step process which involves generating its own

labeled training data and produces a trained classifier that can be used by the extractor. The learner uses dependency parsers and syntactic parsers as part of this process. The extractor then parses web pages and applies the classifier to each sentence containing entity pairs to detect and identify relations. Finally, the assessor tries to assign confidence on each extracted relation based on probability scores derived from evidence and thresholds weak relations.

Surdeanu *et al.* (2012) proposed MIMLRE which employs distant supervision to train a model for extracting relations between two entities. The basic idea is to use existing structured data sources such as Freebase, DbPedia to identify entity pairs that are involved in different relations of interest. Then, we extract candidate sentences by crawling the crawling with a heuristic that any sentence containing the entity pairs identified in previous step describe the relationship between two entities. Now, using these labelled candidate sentences, Surdeanu *et al.* (2012) train a multi label classifier that is capable of assigning more than one relation for a given entity pair. For instance, in the case of the tuple $(BarackObama, UnitedStates)$, two types are relations exist namely $employedBy$ and $bornIn$. While most of the previous methods deal with extracting only one type of relation, MIMLRE model identifies and assigns more than one relation type for a given entity pair.

Overall, all systems discussed in this section have parameters which require configuration based on the domain on which they are applied. While these parameters are defined clearly, there is no discussion on how to choose optimal values for these parameters based on a given experimental setting. The values are mostly hard coded and selected only by experience.

5.1.3 Unsupervised Methods

Much of the research on relation extraction and detection has focussed on applying supervised methods of machine learning. Some techniques discussed above employ seed data to achieve the same in a semi-supervised manner. In this section, we review some systems that employ unsupervised algorithms such as clustering (Grira *et al.*, 2004) and topic modeling (Blei, 2012) to perform relation extraction without using any seed data. Yan *et al.* (2009) and Gonzalez and Turmo (2009) employ clustering to group entity pairs that belong to same relation tuple. de Lacalle and Lapata (2013) take a different approach and uses topic modeling with First order logic to perform relation extraction.

Yan *et al.* (2009) exploit the implicit structural organization of Wikipedia articles to derive entity pairs that are related. That is, they use anchor links from a given entity page to identify target entity. With the derived entity pairs, they identify candidate sentences from snippets returned by search engines and use them to extract three kinds of useful information: (1) ranked relation terms (2) surface patterns and (3) dependency patterns. Ranked relation terms are indicators of particular relation types between entity pairs. Surface patterns are derived from sentences by classifying words between entities as content and functional words. Dependency patterns are derived from the dependency path between the two entities. First, using k-means clustering algorithm and a custom defined distance function, entity pairs are clustered using dependency patterns. Then, another level of clustering is performed using surface patterns since dependency patterns could be restrictive. Finally, the ranked relation labels are used to assign labels for the grouped clusters.

Gonzalez and Turmo (2009) use two components, *scorer* and *filterer* to perform clustering on entity tuples. They identify candidates using target instant types from a training corpus and extract distance and POS features from the candidate sentences. These instances are clustered using probabilistic generative clustering models. Then, they are scored for quality based on size and homogeneousness. The filterer is a simple boundary classifier that is used to identify if a given instance belongs to a cluster or not. It learns a threshold to do this classification by choosing the point of maximum compression of the instance set. Now using the filterer, each instance from the test set is assigned to a particular relation cluster while taking account of relative quality of the clusters from scorer.

de Lacalle and Lapata (2013) combine relational LDA (Yao *et al.*, 2011) with *first order logic* (FOL) rules to build a system for relation extraction. In relational LDA, each document is a mixture of relations over tuples. They use features including the source entity, destination entity, dependency path, entity types and trigger words. They interface FOL rules with a relational LDA model to perform clustering. Using two kinds of auto generated rules from a training corpus namely must-link and cannot-link tuple rules, they produce clusters for each target relation type from the entity pairs.

5.2 Slot Filling Systems

From the specification of the KBP English Slot Filling (ESF) Task, it can be observed that ESF task is same as relation extraction with some added constraints. In the ESF task, the source entity types are restricted to be one of the following two

types: (1)PERSON (PER) (2) ORGANIZATION (ORG). Similarly, in the case of ESF task, the target relation types are defined and fixed. There are 25 relation types for PERSON query entities and 16 relation types for ORGANIZATION query entities. No other relation types and corresponding entities are expected to be extracted in this task. These restrictions make ESF task less challenging compared to relation extraction in general and makes the problem more defined. Hence, it has generated great interest in building systems specifically for the ESF task and several such systems have been participating in the KBP ESF task track during the past few years. In this section, we will highlight few different approaches taken to built Slot Filling systems taking its specifications into account.

In 2014, Stanford's slot filling system (Angeli *et al.*, 2014) based on the DeepDive framework (Niu *et al.*, 2012) was the top performing system in KBP ESF task. DeepDive works in three phases: (1) Feature extraction (2) Engineered constraints (3) Statistical inference and learning. Candidate sentences are extracted by distant supervision and their respective entity and relation mentions are derived. Using this, a factor graph is constructed and Gibbs sampler is run to obtain relation probabilities. They use Freebase as their source for obtaining distantly supervised training data. Their system obtained an F1 score of 36.7 on the official KBP ESF evaluation.

RPI_Blender (Hong *et al.*, 2014) is another slot filling system that employs *Multidimensional truth finding* (Yu *et al.*, 2013) to filter candidate sentences that may express potential relations between target entities. To search for entities, they use their Apache Lucene based search engine with fuzzy matching techniques. Us-

ing co occurrence frequency, they obtain top terms to perform query expansion which helps increase the relevance and reduces noise of candidate sentences. Then, justifications are extracted from the candidate sentences and supplied to their proposed truth finding model. The truth finding model builds a knowledge graph using entities, relations and multiple sources from the corpus. They use several heuristic measures to assign credibility scores for the identified relations in this graph. In addition, they also encode several hard constraints such as entity types based on relation types and soft constraints such as informativeness as part of this graph to further refine the identified relations. Using these signals as features, they build a supervised SVM classifier to finally determine if a relation is true or false. The RPI_Blender system achieved an F1 score of 34.07 in the official KBP 2014 evaluation.

The BLP-TI system from University of Texas at Austin (Bentor *et al.*, 2013) is built on top of Saarland Universitys *RelationFactory* system (Roth *et al.*, 2014a) to explore the impact of their textual inference schemes in the Slot Filling task. The pipeline of the RelationFactory system comprises of two stages: (1) candidate generation, where relevant documents are retrieved and sentences that might have possible relations expressed are filtered using entity type checking (2) candidate validation, where an SVM classifier is used to determine if candidate sentences express a valid relation for the query entities. Their online rule learning system (Raghavan and Mooney, 2013) is used to do learn first order logic (FOL) rules from the set of identified relations. In the inference stage, these rules are used to generate additional relations from the candidate relations extracted directly from the text.

The BLP-TI system achieved an F1 score of 29.15 in the KBP ESF task official evaluation.

The slot filling system from University of Massachusetts known as UMass_IESL (Roth *et al.*, 2014b) applies matrix factorization to perform relation prediction. They also use distant supervision to identify candidate sentences. Relation prediction by Universal Schema Matrix Factorization (Riedel *et al.*, 2013) extracts entity pairs, relations and context between entities from these candidate sentences and leverages the implicit co occurrence information. Each row in the schema matrix represents an entity and each column represents a relation type. They employ factorization based on logistic regression (Collins *et al.*, 2001) for the given matrix construction. Their system was able to achieve an F1 score of 27.39 in the official KBP ESF evaluation.

We can observe that different systems employ different techniques to perform slot filling. As a result, these systems have different strengths and capabilities that prove useful when combined.

5.3 Ensemble Systems

In this section, we review the application of ensemble learning algorithms for different NLP problems. We also look at ensemble systems specifically developed for Relation Extraction and Slot Filling in the past.

5.3.1 Applications in NLP

All these principles for designing ensemble systems have been applied to ensemble NLP systems in different problem domains. Let us look at few such ensemble systems used for language problems in this section.

Henderson and Brill (1998) build an ensemble system for parsing treebanks which performs better than standalone parser. They try two approaches to hybridize their parses. First, they pick the constituents of a parse by majority voting. Second, they try to build a naive bayes classifier that learns which constituents should be included in the parse. Effectively, the classifier determines how much each parser should be trusted. Their ensemble parsing system uses three individual parsers and performs better than the best individual parsing system.

Pedersen (2000) build an ensemble system for word sense disambiguation task. They use a variation of bag of words method using binary features to represent context in both left and right directions of the target word to be disambiguated. They choose 9 different window sizes for both left and right directions: $0, 1, 2, 3, 4, 5, 10, 25, 50$ words. For every combination of left and right window size, a naive Bayes classifier is trained. This step produces 81 candidate naive bayes classifiers. These models are grouped into three broad categories namely (1) narrow ($0, 1, 2$ words) (2) medium ($3, 4, 5, 10$ words) (3) wide ($25, 50$ words). For each target word, the top performing classifiers in each of the nine combinations (left & right) of these categories is picked to participate in the ensemble system. Finally, the sense that receives maximum votes is used to disambiguate the target word. This approach improves the accuracy for disambiguating nouns *line* and *in-*

terest by 1% and 2% respectively.

Whitehead and Yaeger (2010) apply standard ensemble learning algorithms to perform sentiment analysis from text. The ensemble methods they tried include bagging, boosting, random subspace and bagging random subspaces. They choose to use Support Vector Machines (SVM) as their base classifier. For bagging, they used 50 subsets of training data to produce 50 models. In ADABOOST, they chose 50 iterations for the training process. They used K-fold cross validation with 10-folds to compare the results of different methods. They observed that bagging random subspaces model performs better than other methods.

5.3.2 Ensembling RE Systems

Google proposed their Knowledge Vault system (Dong *et al.*, 2014) to aggregate information from different sources across the web and build a knowledge base. They employ different relation extractors for a variety of information sources such as text documents, HTML tables, HTML trees and human annotated pages across the Web. They fuse the output of these extractors by building a binary meta-classifier. The feature vector for this classifier consists of the following information from individual extractors to be fused: (1) number of source documents (2) mean confidence score from different sources. They choose to use a boosted decision stump for their final binary classifier (Reyzin and Schapire, 2006). This work demonstrates a mechanism for combining the output of heterogeneous relation extractors.

Sigletos *et al.* (2005) use stacking (Wolpert, 1992) to combine the output of Information Extraction (IE) systems. This addresses the situation where most systems could be wrong in which case majority voting would fail. They propose the construction of a merged template which combines the output of different IE systems. They have designated representation for extractions that are missing in some IE systems. Using this merged template, they train a meta-classifier which finally decides which extractions are correct. They show that disagreement between different IE systems are exploited by stacking them and also report improved performance.

5.3.3 Ensembling SF Systems

All approaches described above could also be used to build ensemble systems for KBP English Slot Filling (ESF) task. As described before, the goal of KBP Slot Filler Validation (SFV) task is to combine the output from multiple slot filling systems or use language processing to validate the fillers from those systems. The objective here is to maximize the F1-score. Several ensemble systems have been proposed to combine results of slot filling. Let us look at different approaches used to ensemble slot filling systems.

Some systems use rule based consistency checks to perform filler validation. Angeli *et al.* (2013) employ such a scheme using weighted constraint satisfaction problem (CSP). The constraints are hand engineered and establish rules either on specific slots or between pairs of slots or even global constraints. Here, the objective of constraint satisfaction is to maximize the sum of confidences of fillers subject to

the hand engineered constraints on slots. Two classes of rules are used: (1) Filter rules (2) Rewrite rules. Examples of useful filter rules could be *Filter slot fillers that do not match entity type for a relation* or Filter slot fillers that exceed particular length. Example of a rewrite rule could be rewrite filler with its canonical mention as determined by coreferencing. Such rules are helpful in combining results from different slot filling systems.

Cheng *et al.* (2013) pose SFV as Recognizing Textual Entailment (RTE) task and uses rules to check if a relation is satisfied by the (query,filler) pair reported by the slot filling systems. They preprocess the entire ESF task corpus using tokenizer, POS tagger and NER tool as the first step. Similar to previous work on consistency checking, they employ filter rules for slot fillers based on entity types of arguments for different relation types. In addition, they add hand engineered rules to do relation matching, a task that makes sure if a relation is entailed by a candidate sentence and its arguments. These rules specify lexical patterns and enforce positional constraints for entities (query and filler) based on the type of relation. By doing error analysis, they have identified and added around 600 such rules which form the core of their SFV system.

In another work, Sammons *et al.* (2014) develop a trust based SFV system. The principle here is to use the source of slot filler as valuable information for evaluating the correctness of it. They employ majority voting to achieve this. They use learned threshold values on the number of systems that extract a particular filler to decide if it must be included in the result. In general, the threshold values increase with decreasing frequency of the filler. For instance, the most frequent filer is in-

cluded if it occurs at least 5 times while the second most frequent filler is included only if it occurs at least 10 times. When thresholds are high, the system tends to produce high precision, low recall behavior as expected. They also filter fillers based on offset checks using the offsets reported by individual slot filling systems.

Wang *et al.* (2013) attempt to use the confidence values returned by individual SF systems with each extracted filler. They use constrained optimization to aggregate the individual confidence values and produce a single aggregated confidence score. This score is used to determine if the filler must be included in the final result. They apply this scheme to both single valued and list valued slots. In addition, they also explore the possibility of learning weights for individual SF systems based on their overall performance and use them in computing aggregated confidence scores.

Though several systems have been developed to ensemble SF systems, none of them use stacking to combine the results of individual SF systems. From our literature survey, we observe that we are the first to employ Stacking for KBP English Slot Filling and Slot Filler Validation tasks. This makes our contribuition unique and the results from our experiments only increase the significance of Stacking in ensembling SF systems.

Chapter 6

Future Work and Conclusion

In this section, we discuss possible directions of future work in the area of ensembling slot filling systems and conclude this thesis report.

6.1 Future Work

The problem of ensembling slot filling systems is not fully explored. The system proposed in this report uses only confidence score and slot type fields to ensemble the output of SF systems. Many other systems developed for SFV task also use basic ensembling techniques like voting Sammons *et al.* (2014) based on slot filler strings.

There are two strong motivations to explore further into ensembling SF systems. First, ESF is a challenging task since the best performing system achieves F1 score of 40 only. Second, it is hard to develop ESF systems from scratch. Each system involves several components and a complex pipeline. Hence, further exploration in the lines of ensembling existing systems could lead to promising improvements in relatively shorter timeframe. In this section, we discuss some ideas that could be explored in future to improve the system proposed in this report and in general to ensemble SF systems.

It would be interesting to investigate adding more features to ensemble SF system. The output from SF systems contain two types of provenances: (1) relation

provenance (2) filler provenance. Relation provenance provides the offset of text in the source document that describes the relation between the query entity and filler entity. Filler provenance provides the offset of filler entity. Many SFV systems use provenance information to perform consistency checks and filter mechanisms. But, it is possible to design features from these provenances that capture the agreement or disagreement between SF systems in a more fine grained fashion (Viswanathan *et al.*, 2015). For instance, slot fillers can be assigned ranks based on the extent of agreement in their provenances. Also, we can use these provenances to identify the number of unique sources for a particular filler text and use it as a feature. More detailed research could lead to better engineered features.

Alias information could also be used to engineer new features for the ensemble classifier. If a number of SF systems extract the same entity for a $(query, slot)$ pair, it indicates their agreement. After alias merging, we know the number of systems that extracted the same entity for a $(query, slot)$ pair. We can use this to again compute a score that measure agreement at filler entity level. These additional provenance and alias features could strengthen the performance of stacking based Ensemble SF system detailed in this report.

Another promising direction is to explore different ensemble methods for combining the individual SF systems. Sammons *et al.* (2014) used Voting for ensembling SF systems. We are the first to use Stacking for ensembling SF systems. There are several other methods that can be readily tried with SF systems. If there is sufficient training data, we could try extending the stacked meta-classifier for each slot type instead of using one for all slot types. At the moment, the data from KBP

ESF 2013 and 2014 is not sufficient to train a separate meta-classifier for each slot type. Another method is to use subsets of features to train different ensemble classifiers and then use voting to determine final predictions. All these ideas could be explored to understand the relative performance of different methods in ensembling SF systems.

6.2 Conclusion

Relation Extraction is a significant problem that is useful to automatically populating structured knowledge bases. TAC conducts English Slot Filling (ESF) and Slot Filler Validation (SFV) tasks as part of its Knowledge Base Population (KBP) track to promote research in this area. This thesis dealt with developing an Ensembled SF system.

The contributions of this thesis can be summarized as follows. We are the first to use Stacking to develop an Ensemble SF system. We use confidence scores from multiple SF systems and slot type information for our stacked meta-classifier. We have demonstrated through our experiments that stacking is a promising approach and further establish new state-of-the-art results on TAC KBP ESF task. Our stacked meta-classifier provides an F1 score of 47 on KBP ESF task and performs better than the top performing systems in both ESF and SFV tasks (2014). We have developed a scheme by inverting a query expansion technique and used it to merge alias extractions from different SF systems. From our experimental results, we observe that this scheme did not have an impact on the performance of the Ensembled SF system.

In future, additional information from output of SF systems such as provenance, alias fillers, query entity type and filler entity type can be used to engineer more features for the Ensemble classifier. Also, other ensemble learning methods such as bagging can be applied for the task of Ensembling SF systems.

The promising results from our stacked Ensemble SF system indicate that ensembling boosts the performance of SF systems to a good extent. It is known that Slot Filling is a challenging task since the best performing individual SF system achieves only an F1 score of 39. Also, developing new SF systems is time consuming due to the numerous pre processing steps and complex components involved in the pipeline. These facts coupled with the results presented in this thesis report indicate that exploring ensembling methods using existing SF systems might produce better performing SF systems in a shorter time frame.

References

Eugene Agichtein and Luis Gravano. Snowball: Extracting relations from large plain-text collections. In *Proceedings of the fifth ACM conference on Digital libraries*, pages 85–94. ACM, 2000.

Gabor Angeli, Arun Chaganty, Angel Chang, Kevin Reschke, Julie Tibshirani, Jean Y Wu, Osbert Bastani, Keith Siilats, and Christopher D Manning. Stanfords 2013 kbp system, 2013.

Gabor Angeli, Sonal Gupta, Melvin Jose, Christopher D Manning, Christopher Ré, Julie Tibshirani, Jean Y Wu, Sen Wu, and Ce Zhang. Stanfords 2014 slot filling systems. *Proceedings of the Seventh Text Analysis Conference (TAC 2014)*, 2014.

Nguyen Bach and Sameer Badaskar. A review of relation extraction. *Literature review for Language and Statistics II*, 2007.

Michele Banko, Michael J Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. Open information extraction for the web. In *IJCAI*, volume 7, pages 2670–2676, 2007.

Yinon Bentor, Amelia Harrison, Shruti Bhosale, and Raymond Mooney. University of texas at austin kbp 2013 slot filling system: Bayesian logic programs for textual inference. In *Text Analysis Conference (TAC 2013)*, 2013.

David M Blei. Probabilistic topic models. *Communications of the ACM*, 55(4):77–84, 2012.

Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100. ACM, 1998.

Kurt Bollacker, Robert Cook, and Patrick Tufts. Freebase: A shared database of structured general human knowledge. In *AAAI*, volume 7, pages 1962–1963, 2007.

Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

Sergey Brin. Extracting patterns and relations from the world wide web. In *The World Wide Web and Databases*, pages 172–183. Springer, 1999.

Razvan C Bunescu and Raymond J Mooney. A shortest path dependency kernel for relation extraction. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 724–731. Association for Computational Linguistics, 2005.

Razvan C Bunescu and Raymond J Mooney. Subsequence kernels for relation extraction. In *Advances in neural information processing systems*, pages 171–178, 2005.

Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.

Xiao Cheng, Bingling Chen, Rajhans Samdani, Kai-Wei Chang, Zhiye Fei, Mark Sammons, John Wieting, Subhro Roy, Chizheng Wang, and Dan Roth. Illinois cognitive computation group ui-ccg tac 2013 entity linking and slot filler validation systems. 2013.

Michael Collins, Sanjoy Dasgupta, and Robert E Schapire. A generalization of principal components analysis to the exponential family. In *Advances in neural information processing systems*, pages 617–624, 2001.

Aron Culotta and Jeffrey Sorensen. Dependency tree kernels for relation extraction. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 423. Association for Computational Linguistics, 2004.

Oier Lopez de Lacalle and Mirella Lapata. Unsupervised relation extraction with general domain knowledge. In *EMNLP*, pages 415–425, 2013.

Thomas G Dietterich. Ensemble methods in machine learning. In *Multiple classifier systems*, pages 1–15. Springer, 2000.

Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 601–610. ACM, 2014.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874, 2008.

Yoav Freund, Robert E Schapire, et al. Experiments with a new boosting algorithm. In *ICML*, volume 96, pages 148–156, 1996.

Edgar Gonzalez and Jordi Turmo. Unsupervised relation extraction by massive clustering. In *Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on*, pages 782–787. IEEE, 2009.

Nizar Grira, Michel Crucianu, and Nozha Boujemaa. Unsupervised and semi-supervised clustering: a brief survey. *A review of machine learning techniques for processing multimedia content, Report of the MUSCLE European Network of Excellence (FP6)*, 2004.

Zhou GuoDong, Su Jian, Zhang Jie, and Zhang Min. Exploring various knowledge in relation extraction. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 427–434. Association for Computational Linguistics, 2005.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18, 2009.

Lars Kai Hansen and Peter Salamon. Neural network ensembles. *IEEE transactions on pattern analysis and machine intelligence*, 12(10):993–1001, 1990.

John C Henderson and Eric Brill. Exploiting diversity for natural language processing: Combining parsers. In *AAAI/IAAI*, page 1174, 1998.

Yu Hong, Xiaobin Wang, Yadong Chen, Jian Wang, Tongtao Zhang, Jin Zheng, Dian Yu, Qi Li, Boliang Zhang, Han Wang, et al. Rpi blender tac-kbp2014 knowledge base population system. 2014.

Nanda Kambhatla. Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, page 22. Association for Computational Linguistics, 2004.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, et al. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 2014.

Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. Text classification using string kernels. *The Journal of Machine Learning Research*, 2:419–444, 2002.

Feng Niu, Ce Zhang, Christopher Ré, and Jude W Shavlik. Deepdive: Web-scale knowledge-base construction using statistical learning and inference. *VLDS*, 12:25–28, 2012.

Ted Pedersen. A simple approach to building ensembles of naive bayesian classifiers for word sense disambiguation. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pages 63–69. Association for Computational Linguistics, 2000.

Sindhu Raghavan and Raymond J Mooney. Online inference-rule learning from natural-language extractions. In *AAAI Workshop: Statistical Relational Artificial Intelligence*, 2013.

Lev Reyzin and Robert E Schapire. How boosting the margin can also boost classifier complexity. In *Proceedings of the 23rd international conference on Machine learning*, pages 753–760. ACM, 2006.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. Relation extraction with matrix factorization and universal schemas. *The Selected-Works of Benjamin M. Marlin*, 2013.

Benjamin Roth and Dietrich Klakow. Cross-language retrieval using link-based language models. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 773–774. ACM, 2010.

Benjamin Roth, Tassilo Barth, Michael Wiegand, Mittul Singh, and Dietrich Klakow. Effective slot filling based on shallow distant supervision methods. *arXiv preprint arXiv:1401.1158*, 2014.

Benjamin Roth, Emma Strubell, John Sullivan, Lakshmi Vikraman, et al. Universal schema for slot-filling, cold-start kbp and event argument extraction: Umass iesl at tac kbp 2014. 2014.

Mark Sammons, Yangqiu Song, Ruichen Wang, Gourab Kundu, et al. Overview of UI-CCG systems for event argument extraction, entity discovery and linking, and slot filler validation. *Proceedings of the Seventh Text Analysis Conference (TAC2014)*, 2014.

Georgios Sigletos, Georgios Paliouras, Constantine D Spyropoulos, and Michalis Hatzopoulos. Combining information extraction systems using voting and stacked generalization. *The Journal of Machine Learning Research*, 6:1751–1782, 2005.

Joseph Sill, Gábor Takács, Lester Mackey, and David Lin. Feature-weighted linear stacking. *arXiv preprint arXiv:0911.0460*, 2009.

Mihai Surdeanu and Heng Ji. Overview of the english slot filling track at the tac2014 knowledge base population evaluation. In *Proceedings of the Seventh Text Analysis Conference (TAC 2014)*, 2014.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D Manning. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 455–465. Association for Computational Linguistics, 2012.

Mihai Surdeanu. Overview of the tac2013 knowledge base population evaluation: English slot filling and temporal slot filling. In *Proceedings of the Sixth Text Analysis Conference (TAC 2013)*, 2013.

Vidhoon Viswanathan, Nazneen Fatema N Rajani, Yinon Bentor, and Raymond J Mooney. Stacked ensembles of information extractors for knowledge base population. In *Proceedings of the 53rd annual meeting on association for computational linguistics*. Association for Computational Linguistics, 2015.

I-Jeng Wang, Edwina Liu, Cash Costello, and Christine Piatko. Jhuapl tac-kbp2013 slot filler validation system. 2013.

Matthew Whitehead and Larry Yaeger. Sentiment mining using ensemble classification models. In *Innovations and advances in computer sciences and engineering*, pages 509–514. Springer, 2010.

David H Wolpert. Stacked generalization. *Neural networks*, 5(2):241–259, 1992.

Yulan Yan, Naoaki Okazaki, Yutaka Matsuo, Zhenglu Yang, and Mitsuru Ishizuka. Unsupervised relation extraction by mining wikipedia texts using information from the web. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1021–1029. Association for Computational Linguistics, 2009.

Limin Yao, Aria Haghighi, Sebastian Riedel, and Andrew McCallum. Structured relation discovery using generative models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1456–1466. Association for Computational Linguistics, 2011.

David Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, pages 189–196. Association for Computational Linguistics, 1995.

Dian Yu, Haibo Li, T Cassidy, Q Li, H Huang, Z Chen, H Ji, Y Zhang, and D Roth. Rpi-blender tac-kbp2013 knowledge base population system. In *Proceedings of the Sixth Text Analysis Conference*, 2013.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. Kernel methods for relation extraction. *The Journal of Machine Learning Research*, 3:1083–1106, 2003.

Shubin Zhao and Ralph Grishman. Extracting relations with integrated information using kernel methods. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 419–426. Association for Computational Linguistics, 2005.

Vita

Vidhoon Viswanathan was born in Coimbatore, India on July 5, 1990, the only son of Mr. Viswanathan and Mrs. Uma Viswanathan. He completed higher secondary school education at R.G. Matriculation Higher Secondary School, Udumalpet. In 2007, he joined College of Engineering, Guindy (Anna University), Chennai to pursue his Bachelor of Technology degree in Information Technology. In 2011, after completing his undergraduate studies, he joined Texas Instruments (India) Ltd. and worked in the OMAP business unit. In early 2013, he quit his job and traveled around North India for a couple of months. In August 2013, he joined the Graduate School at The University of Texas at Austin to pursue his Master of Science degree in Computer Science.

Permanent Address: 11, Jothi Nagar,
S.V.Mills Post,
Udumalpet, Tiruppur District,
Tamilnadu, India - 642128.

This thesis was typeset with LaTeX 2_ε[1] by the author.

[1] LaTeX 2_ε is an extension of LaTeX. LaTeX is a collection of macros for TeX. TeX is a trademark of the American Mathematical Society. The macros used in formatting this thesis were written by Dinesh Das, Department of Computer Sciences, The University of Texas at Austin, and extended by Bert Kay, James A. Bednar, and Ayman El-Khashab.