Automatic Creation of Domain Templates

Elena Filatova[*], **Vasileios Hatzivassiloglou**[†] **and Kathleen McKeown**[*]

[*]Department of Computer Science
Columbia University
{filatova,kathy}@cs.columbia.edu

[†]Department of Computer Science
The University of Texas at Dallas
vh@hlt.utdallas.edu

Abstract

Recently, many Natural Language Processing (NLP) applications have improved the quality of their output by using various machine learning techniques to mine Information Extraction (IE) patterns for capturing information from the input text. Currently, to mine IE patterns one should know in advance the type of the information that should be captured by these patterns. In this work we propose a novel methodology for corpus analysis based on cross-examination of several document collections representing different instances of the same domain. We show that this methodology can be used for automatic domain template creation. As the problem of automatic domain template creation is rather new, there is no well-defined procedure for the evaluation of the domain template quality. Thus, we propose a methodology for identifying what information should be present in the template. Using this information we evaluate the automatically created domain templates through the text snippets retrieved according to the created templates.

1 Introduction

Open-ended question-answering (QA) systems typically produce a response containing a variety of specific facts proscribed by the question type. A biography, for example, might contain the date of birth, occupation, or nationality of the person in question (Duboue and McKeown, 2003; Zhou et al., 2004; Weischedel et al., 2004; Filatova and Prager, 2005). A definition may contain the genus of the term and characteristic attributes (Blair-Goldensohn et al., 2004). A response to a question about a terrorist attack might include the event, victims, perpetrator and date as the templates designed for the Message Understanding Conferences (Radev and McKeown, 1998; White et al., 2001) predicted. Furthermore, the type of information included varies depending on context. A biography of an actor would include movie names, while a biography of an inventor would include the names of inventions. A description of a terrorist event in Latin America in the eighties is different from the description of today's terrorist events.

How does one determine what facts are important for different kinds of responses? Often the types of facts that are important are hand encoded ahead of time by a human expert (e.g., as in the case of MUC templates). In this paper, we present an approach that allows a system to learn the types of facts that are appropriate for a particular response. We focus on acquiring fact-types for events, automatically producing a *template* that can guide the creation of responses to questions requiring a description of an event. The template can be tailored to a specific time period or country simply by changing the document collections from which learning takes place.

In this work, a *domain* is a set of events of a particular type; earthquakes and presidential elections are two such domains. Domains can be instantiated by several *instances* of events of that type (e.g., the earthquake in Japan in October 2004, the earthquake in Afghanistan in March 2002, etc.).[1] The granularity of domains and instances can be altered by examining data at different levels of detail, and domains can be hierarchically structured. An ideal template is a set of attribute-value pairs, with the attributes specifying particular functional roles important for the domain events.

In this paper we present a method of domain-independent on-the-fly template creation. Our method is completely automatic. As input it requires several document collections describing domain instances. We cross-examine the input instances, we identify verbs important for the majority of instances and relationships containing these verbs. We generalize across multiple domain instances to automatically determine which of these relations should be used in the template. We report on data collection efforts and results from four domains. We assess how well the automatically produced templates satisfy users' needs, as manifested by questions collected for these domains.

[1]Unfortunately, NLP terminology is not standardized across different tasks. Two NLP tasks most close to our research are Topic Detection and Tracking (TDT) (Fiscus et al., 1999) and Information Extraction (IE) (Marsh and Perzanowski, 1997). In TDT terminology, our domains are topics and our instances are events. In IE terminology, our domains are scenarios and our domain templates are scenario templates.

2 Related Work

Our system automatically generates a template that captures the generally most important information for a particular domain and is reusable across multiple instances of that domain. Deciding what slots to include in the template, and what restrictions to place on their potential fillers, is a knowledge representation problem (Hobbs and Israel, 1994). Templates were used in the main IE competitions, the Message Understanding Conferences (Hobbs and Israel, 1994; Onyshkevych, 1994; Marsh and Perzanowski, 1997). One of the recent evaluations, ACE,[2] uses pre-defined frames connecting event types (e.g., *arrest*, *release*) to a set of attributes. The template *construction* task was not addressed by the participating systems. The domain templates were created manually by experts to capture the structure of the facts sought.

Although templates have been extensively used in information extraction, there has been little work on their automatic design. In the Conceptual Case Frame Acquisition project (Riloff and Schmelzenbach, 1998), extraction patterns, a domain semantic lexicon, and a list of conceptual roles and associated semantic categories for the domain are used to produce multiple-slot case frames with selectional restrictions. The system requires two sets of documents: those relevant to the domain and those irrelevant. Our approach does not require any domain-specific knowledge and uses only corpus-based statistics.

The GISTexter summarization system (Harabagiu and Maiorano, 2002) used statistics over an arbitrary document collection together with semantic relations from WordNet. The created templates heavily depend on the topical relations encoded in WordNet. The template models an input collection of documents. If there is only one domain instance described in the input than the template is created for this particular instance rather than for a domain. In our work, we learn domain templates by cross-examining several collections of documents on the same topic, aiming for a general domain template. We rely on relations cross-mentioned in different instances of the domain to automatically prioritize roles and relationships for selection.

Topic Themes (Harabagiu and Lăcătuşu, 2005) used for multi-document summarization merge various arguments corresponding to the same semantic roles for the semantically identical verb phrases (e.g., *arrests* and *placed under arrest*).

Atomic events also model an input document collection (Filatova and Hatzivassiloglou, 2003) and are created according to the statistics collected for co-occurrences of named entity pairs linked through actions. GISTexter, atomic events, and Topic Themes were used for modeling a collection of documents rather than a domain.

In other closely related work, Sudo et al. (2003) use frequent dependency subtrees as measured by TF*IDF to identify named entities and IE patterns important for a given domain. The goal of their work is to show how the techniques improve IE pattern acquisition. To do this, Sudo et al. constrain the retrieval of relevant documents for a MUC scenario and then use unsupervised learning over descriptions within these documents that match specific types of named entities (e.g., *Arresting Agency, Charge*), thus enabling learning of patterns for specific templates (e.g., the Arrest scenario). In contrast, the goal of our work is to show how similar techniques can be used to learn what information is important for a given domain or event and thus, should be included into the domain template. Our approach allows, for example, learning that an arrest along with other events (e.g., attack) is often part of a terrorist event. We do not assume any prior knowledge about domains. We demonstrate that frequent subtrees can be used not only to extract specific named entities for a given scenario but also to learn domain-important relations. These relations link domain actions and named entities as well as general nouns and words belonging to other syntactic categories.

Collier (1998) proposed a fully automatic method for creating templates for information extraction. The method relies on Luhn's (1957) idea of locating statistically significant words in a corpus and uses those to locate the sentences in which they occur. Then it extracts Subject-Verb-Object patterns in those sentences to identify the most important interactions in the input data. The system was constructed to create MUC templates for *terrorist attacks*. Our work also relies on corpus statistics, but we utilize arbitrary syntactic patterns and explicitly use multiple domain instances. Keeping domain instances separated, we cross-examine them and estimate the importance of a particular information type in the domain.

[2] `http://www.nist.gov/speech/tests/ace/index.htm`

3 Our Approach to Template Creation

After reading about presidential elections in different countries on different years, a reader has a general picture of this process. Later, when reading about a new presidential election, the reader already has in her mind a set of questions for which she expects answers. This process can be called *domain modeling*. The more instances of a particular domain a person has seen, the better understanding she has about what type of information should be expected in an unseen collection of documents discussing a new instance of this domain.

Thus, we propose to use a set of document collections describing different instances within one domain to learn the general characteristics of this domain. These characteristics can be then used to create a domain template. We test our system on four domains: airplane crashes, earthquakes, presidential elections, terrorist attacks.

4 Data Description

4.1 Training Data

To create training document collections we used BBC Advanced Search[3] and submitted queries of the type ⟨*domain title* + *country*⟩. For example, ⟨“presidential election” USA⟩.

In addition, we used BBC's Advanced Search date filter to constrain the results to different date periods of interest. For example, we used known dates of elections and allowed a search for articles published up to five days before or after each such date. At the same time for the terrorist attacks or earthquakes domain the time constraints we submitted were the day of the event plus ten days.

Thus, we identify several instances for each of our four domains, obtaining a document collection for *each* instance. E.g., for the earthquake domain we collected documents on the earthquakes in Afghanistan (March 25, 2002), India (January 26, 2001), Iran (December 26, 2003), Japan (October 26, 2004), and Peru (June 23, 2001). Using this procedure we retrieve training document collections for 9 instances of airplane crashes, 5 instances of earthquakes, 13 instances of presidential elections, and 6 instances of terrorist attacks.

4.2 Test Data

To test our system, we used document clusters from the Topic Detection and Tracking (TDT) corpus (Fiscus et al., 1999). Each TDT topic has a topic label, such as *Accidents* or *Natural Disasters*.[4] These categories are broader than our domains. Thus, we manually filtered the TDT topics relevant to our four training domains (e.g., Accidents matching Airplane Crashes). In this way, we obtained TDT document clusters for 2 instances of airplane crashes, 3 instances of earthquakes, 6 instances of presidential elections and 3 instances of terrorist attacks. The number of the documents corresponding to the instances varies greatly (from two documents for one of the earthquakes up to 156 documents for one of the terrorist attacks). This variation in the number of documents per topic is typical for the TDT corpus. Many of the current approaches of domain modeling collapse together different instances and make the decision on what information is important for a domain based on this generalized corpus (Collier, 1998; Barzilay and Lee, 2003; Sudo et al., 2003). We, on the other hand, propose to cross-examine these instances keeping them separated. Our goal is to eliminate dependence on how well the corpus is balanced and to avoid the possibility of greater impact on the domain template of those instances which have more documents.

5 Creating Templates

In this work we build domain templates around verbs which are estimated to be important for the domains. Using verbs as the starting point we identify semantic dependencies within sentences. In contrast to deep semantic analysis (Fillmore and Baker, 2001; Gildea and Jurafsky, 2002; Pradhan et al., 2004; Harabagiu and Lăcătuşu, 2005; Palmer et al., 2005) we rely only on corpus statistics. We extract the most frequent syntactic subtrees which connect verbs to the lexemes used in the same subtrees. These subtrees are used to create domain templates.

For each of the four domains described in Section 4, we automatically create domain templates using the following algorithm.

Step 1: *Estimate what verbs are important for the domain under investigation.* We initiate our algorithm by calculating the probabilities for all the verbs in the document collection for one domain — e.g., the collection containing all the instances in the domain of airplane crashes. We

[3]`http://news.bbc.co.uk/shared/bsp/search2/advanced/news_ifs.stm`

[4]In our experiments we analyze TDT topics used in TDT-2 and TDT-4 evaluations.

discard those verbs that are stop words (Salton, 1971). To take into consideration the distribution of a verb among different instances of the domain, we normalize this probability by its *VIF* value (verb instance frequency), specifying in how many domain instances this verb appears.

$$Score(vb_i) = \frac{count_{vb_i}}{\sum_{vb_j \in comb\ coll} count_{vb_j}} \times VIF(vb_i) \quad (1)$$

$$VIF(vb_i) = \frac{\#\ of\ domain\ instances\ containing\ vb_i}{\#\ of\ all\ domain\ instances} \quad (2)$$

These verbs are estimated to be the most important for the combined document collection for all the domain instances. Thus, we build the domain template around these verbs. Here are the top ten verbs for the *terrorist attack* domain:

killed, *told*, *found*, *injured*, *reported*,
happened, *blamed*, *arrested*, *died*, *linked*.

Step 2: *Parse those sentences which contain the top 50 verbs.* After we identify the 50 most important verbs for the domain under analysis, we parse all the sentences in the domain document collection containing these verbs with the Stanford syntactic parser (Klein and Manning, 2002).

Step 3: *Identify most frequent subtrees containing the top 50 verbs.* A domain template should contain not only the most important actions for the domain, but also the entities that are linked to these actions or to each other through these actions. The lexemes referring to such entities can potentially be used within the domain template slots. Thus, we analyze those portions of the syntactic trees which contain the verbs themselves plus other lexemes used in the same subtrees as the verbs. To do this we use FREQuent Tree miner.[5] This software is an implementation of the algorithm presented by (Abe et al., 2002; Zaki, 2002), which extracts frequent ordered subtrees from a set of ordered trees. Following (Sudo et al., 2003) we are interested only in the lexemes which are near neighbors of the most frequent verbs. Thus, we look only for those subtrees which contain the verbs themselves and from four to ten tree nodes, where a node is either a syntactic tag or a lexeme with its tag. We analyze not only NPs which correspond to the subject or object of the verb, but other syntactic constituents as well. For example, PPs can potentially link the verb to locations or dates, and we want to include this information into the template. Table 1 contains a sample of subtrees for the *terrorist attack* domain mined from the sentences containing the verb *killed*. The first column of Table 1 shows how many nodes are in the subtree.

[5]http://chasen.org/~taku/software/freqt/

nodes	subtree
8	(SBAR(S(VP(VBD killed)(NP(QP(IN at))(NNS people)))))
8	(SBAR(S(VP(VBD killed)(NP(QP(JJS least))(NNS people)))))
5	(VP(ADVP)(VBD killed)(NP(NNS people)))
6	(VP(VBD killed)(NP(ADJP(JJ many))(NNS people)))
5	(VP(VP(VBD killed)(NP(NNS people))))
7	(VP(ADVP(NP))(VBD killed)(NP(CD 34)(NNS people)))
6	(VP(ADVP)(VBD killed)(NP(CD 34)(NNS people)))

Table 1: Sample subtrees for the *terrorist attack* domain.

Step 4: *Substitute named entities with their respective tags.* We are interested in analyzing a whole domain, not just an instance of this domain. Thus, we substitute all the named entities with their respective tags, and all the exact numbers with the tag NUMBER. We speculate that subtrees similar to those presented in Table 1 can be extracted from a document collection representing any instance of a terrorist attack, with the only difference being the exact number of causalities. Later, however, we analyze the domain instances separately to identity information typical for the domain. The procedure of substituting named entities with their respective tags previously proved to be useful for various tasks (Barzilay and Lee, 2003; Sudo et al., 2003; Filatova and Prager, 2005). To get named entity tags we used BBN's IdentiFinder (Bikel et al., 1999).

Step 5: *Merge together the frequent subtrees.* Finally, we merge together those subtrees which are identical according to the information encoded within them. This is a key step in our algorithm which allows us to bring together subtrees from different instances of the same domain. For example, the information rendered by all the subtrees from the bottom part of Table 1 is identical. Thus, these subtrees can be merged into one which contains the longest common pattern:

(VBD killed)(NP(NUMBER)(NNS people))

After this merging procedure we keep only those subtrees for which each of the domain instances has at least one of the subtrees from the initial set of subtrees. This subtree should be used in the instance at least twice. At this step, we make sure that we keep in the template only the information which is generally important for the domain rather than only for a fraction of instances in this domain. We also remove all the syntactic tags as we want to make this pattern as general for the domain as possible. A pattern without syntactic dependencies contains a verb together with a prospective template slot corresponding to this verb:

killed: (NUMBER) (NNS people)

In the above example, the prospective template slots appear after the verb *killed*. In other cases the domain slots appear in front of the verb. Two examples of such slots, for the *presidential election* and *earthquake* domains, are shown below:

(PERSON) won
(NN earthquake) struck

The above examples show that it is not enough to analyze only named entities, general nouns contain important information as well. We term the structure consisting of a verb together with the associated slots a *slot structure*. Here is a part of the slot structure we get for the verb *killed* after cross-examination of the *terrorist attack* instances:

killed (NUMBER) (NNS people)
(PERSON) killed
(NN suicide) killed

Slot structures are similar to verb frames, which are manually created for the PropBank annotation (Palmer et al., 2005).[6] An example of the PropBank frame for the verb *to kill* is:

Roleset kill.01 "cause to die":
  Arg0:killer
  Arg1:corpse
  Arg2:instrument

The difference between the slot structure extracted by our algorithm and the PropBank frame slots is that the frame slots assign a semantic role to each slot, while our algorithm gives either the type of the named entity that should fill in this slot or puts a particular noun into the slot (e.g., ORGANIZATION, earthquake, people). An ideal domain template should include semantic information but this problem is outside of the scope of this paper.

Step 6: *Creating domain templates*. After we get all the frequent subtrees containing the top 50 domain verbs, we merge all the subtrees corresponding to the same verb and create a slot structure for every verb as described in Step 5. The union of such slot structures created for all the important verbs in the domain is called the *domain template*. From the created templates we remove the slots which are used in all the domains. For example,

(PERSON) told.

□

The presented algorithm can be used to create a template for any domain. It does not require predefined domain or world knowledge. We learn domain templates from cross-examining document collections describing different instances of the domain of interest.

[6] http://www.cs.rochester.edu/~gildea/Verbs/

6 Evaluation

The task we deal with is new and there is no well-defined and standardized evaluation procedure for it. Sudo et al. (2003) evaluated how well their IE patterns captured named entities of three predefined types. We are interested in evaluating how well we capture the major actions as well as their constituent parts.

There is no set of domain templates which are built according to a unique set of principles against which we could compare our automatically created templates. Thus, we need to create a gold standard. In Section 6.1, we describe how the gold standard is created. Then, in Section 6.2, we evaluate the quality of the automatically created templates by extracting clauses corresponding to the templates and verifying how many answers from the questions in the gold standard are answered by the extracted clauses.

6.1 Stage 1. Information Included into Templates: Interannotator Agreement

To create a gold standard we asked people to create a list of questions which indicate what is important for the domain description. Our decision to aim for the lists of questions and not for the templates themselves is based on the following considerations: first, not all of our subjects are familiar with the field of IE and thus, do not necessarily know what an IE template is; second, our goal for this evaluation is to estimate interannotator agreement for capturing the important aspects for the domain and not how well the subjects agree on the template structure.

We asked our subjects to think of their experience of reading newswire articles about various domains.[7] Based on what they remember from this experience, we asked them to come up with a list of questions about a particular domain. We asked them to come up with at most 20 questions covering the information they will be looking for given an unseen news article about a new event in the domain. We did not give them any input information about the domain but allowed them to use any sources to learn more information about the domain.

We had ten subjects, each of which created one list of questions for one of the four domains under

[7] We thank Rod Adams, Cosmin-Adrian Bejan, Sasha Blair-Goldensohn, Cyril Cerovic, David Elson, David Evans, Ovidiu Fortu, Agustin Gravano, Lokesh Shresta, John Yundt-Pacheco and Kapil Thadani for the submitted questions.

	Jaccard metric		
Domain	subj_1 and subj_2 (and subj_3)	subj_1 and MUC	subj_2 and MUC
Airplane crash	0.54	-	-
Earthquake	0.68	-	-
Presidential Election	0.32	-	-
Terrorist Attack	0.50	0.63	0.59

Table 2: Creating gold standard. Jaccard metric values for interannotator agreement.

analysis. Thus, for the *earthquake* and *terrorist attack* domains we got two lists of questions; for the *airplane crash* and *presidential election* domains we got three lists of questions.

After the questions lists were created we studied the agreement among annotators on what information they consider is important for the domain and thus, should be included in the template. We matched the questions created by different annotators for the same domain. For some of the questions we had to make a judgement call on whether it is a match or not. For example, the following question created by one of the annotators for the *earthquake* domain was:

Did the earthquake occur in a well-known area for earthquakes (e.g. along the San Andreas fault), or in an unexpected location?

We matched this question to the following three questions created by the other annotator:

What is the geological localization?
Is it near a fault line?
Is it near volcanoes?

Usually, the degree of interannotator agreement is estimated using Kappa. For this task, though, Kappa statistics cannot be used as they require knowledge of the expected or chance agreement, which is not applicable to this task (Fleiss et al., 1981). To measure interannotator agreement we use the Jaccard metric, which does not require knowledge of the expected or chance agreement. Table 2 shows the values of Jaccard metric for interannotator agreement calculated for all four domains. Jaccard metric values are calculated as

$$Jaccard(domain^d) = \frac{|QS_i^d \cap QS_j^d|}{|QS_i^d \cup QS_j^d|} \tag{3}$$

where QS_i^d and QS_j^d are the sets of questions created by subjects i and j for domain d. For the *airplane crash* and *presidential election* domains we averaged the three pairwise Jaccard metric values.

The scores in Table 2 show that for some domains the agreement is quite high (e.g., *earthquake*), while for other domains (e.g., *presidential election*) it is twice as low. This difference in scores can be explained by the complexity of the domains and by the differences in understanding of these domains by different subjects. The scores for the *presidential election* domain are predictably low as in different countries the roles of presidents are very different: in some countries the president is the head of the government with a lot of power, while in other countries the president is merely a ceremonial figure. In some countries the presidents are elected by general voting while in other countries, the presidents are elected by parliaments. These variations in the domain cause the subjects to be interested in different issues of the domain. Another issue that might influence the interannotator agreement is the distribution of the presidential election process in time. For example, one of our subjects was clearly interested in the pre-voting situation, such as debates between the candidates, while another subject was interested only in the outcome of the presidential election.

For the *terrorist attack* domain we also compared the lists of questions we got from our subjects with the *terrorist attack* template created by experts for the MUC competition. In this template we treated every slot as a separate question, excluding the first two slots which captured information about the text from which the template fillers were extracted and not about the domain. The results for this comparison are included in Table 2.

Differences in domain complexity were studied by IE researchers. Bagga (1997) suggests a classification methodology to predict the syntactic complexity of the domain-related facts. Huttunen et al. (2002) analyze how component subevents of the domain are linked together and discuss the factors which contribute to the domain complexity.

6.2 Stage 2. Quality of the Automatically Created Templates

In section 6.1 we showed that not all the domains are equal. For some of the domains it is much easier to come to a consensus about what slots should be present in the domain template than for others. In this section we describe the evaluation of the four automatically created templates.

Automatically created templates consist of slot structures and are not easily readable by human annotators. Thus, instead of direct evaluation of the template quality, we evaluate the clauses extracted according to the created templates and check whether these clauses contain the answers to the questions created by the subjects during the first stage of the evaluation. We extract the clauses corresponding to the test instances according to the following procedure:

1. Identify all the simple clauses in the documents corresponding to a particular test instance (respective TDT topic). For example, for the sentence

   Her husband, Robert, survived Thursday's explosion in a Yemeni harbor that killed at least six crew members and injured 35.

   only one part is output:

   that killed at least six crew members and injured 35
2. For every domain template slot check all the simple clauses in the instance (TDT topic) under analysis. Find the shortest clause (or sequence of clauses) which includes both the verb and other words extracted for this slot in their respective order. Add this clause to the list of extracted clauses unless this clause has been already added to this list.
3. Keep adding clauses to the list of extracted clauses till all the template slots are analyzed or the size of the list exceeds 20 clauses.

The key step in the above algorithm is Step 2. By choosing the shortest simple clause or sequence of simple clauses corresponding to a particular template slot, we reduce the possibility of adding more information to the output than is necessary to cover each particular slot.

In Step 3 we keep only the first twenty clauses so that the length of the output which potentially contains an answer to the question of interest is not larger than the number of questions provided by each subject. The templates are created from the slot structures extracted for the top 50 verbs. The higher the estimated score of the verb (Eq. 1) for the domain the closer to the top of the template the slot structure corresponding to this verb will be. We assume that the important information is more likely to be covered by the slot structures that are placed near the top of the template.

The evaluation results for the automatically created templates are presented in Figure 1. We calculate what average percentage of the questions is covered by the outputs created according to the domain templates. For every domain, we present the percentage of the covered questions separately for each annotator and for the intersection of questions (Section 6.1).

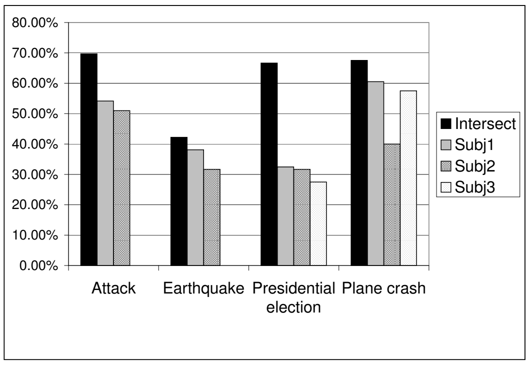

Figure 1: Evaluation results.

For the questions common for all the annotators we capture about 70% of the answers for three out of four domains. After studying the results we noticed that for the *earthquake* domain some questions did not result in a template slot and thus, could not be covered by the extracted clauses. Here are two of such questions:

Is it near a fault line?
Is it near volcanoes?

According to the template creation procedure, which is centered around verbs, the chances that extracted clauses would contain answers to these questions are low. Indeed, only one of the three sentence sets extracted for the three TDT earthquake topics contain an answer to one of these questions.

Poor results for the *presidential election* domain could be predicted from the Jaccard metric value for interannotator agreement (Table 2). There is considerable discrepancy in the questions created by human annotators which can be attributed to the great variation in the *presidential election* domain itself. It must be also noted that most of the questions created for the *presidential election* domain were clearly referring to the democratic election procedure, while some of the TDT topics categorized as *Elections* were about either election fraud or about opposition taking over power without the formal resignation of the previous president.

Overall, this evaluation shows that using automatically created domain templates we extract sentences which contain a substantial part of the important information expressed in questions for that domain. For those domains which have small diversity our coverage can be significantly higher.

7 Conclusions

In this paper, we presented a robust method for data-driven discovery of the important fact-types for a given domain. In contrast to supervised methods, the fact-types are not pre-specified. The resulting slot structures can subsequently be used to guide the generation of responses to questions about new instances of the same domain. Our approach features the use of corpus statistics derived from both lexical and syntactic analysis across documents. A comparison of our system output for four domains of interest shows that our approach can reliably predict the majority of information that humans have indicated are of interest. Our method is flexible: analyzing document collections from different time periods or locations, we can learn domain descriptions that are tailored to those time periods and locations.

Acknowledgements. We would like to thank Rebecca Passonneau and Julia Hirschberg for the fruitful discussions at the early stages of this work; Vasilis Vassalos for his suggestions on the evaluation instructions; Michel Galley, Agustin Gravano, Panagiotis Ipeirotis and Kapil Thadani for their enormous help with evaluation.

This material is based upon work supported in part by the Advanced Research Development Agency (ARDA) under Contract No. NBCHC040040 and in part by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR0011-06-C-0023. Any opinions, findings and conclusions expressed in this material are those of the authors and do not necessarily reflect the views of ARDA and DARPA.

References

Kenji Abe, Shinji Kawasoe, Tatsuya Asai, Hiroki Arimura, and Setsuo Arikawa. 2002. Optimized substructure discovery for semi-structured data. In *Proc. of PKDD*.

Amit Bagga. 1997. Analyzing the complexity of a domain with respect to an Information Extraction task. In *Proc. 7th MUC*.

Regina Barzilay and Lillian Lee. 2003. Learning to paraphrase: An unsupervised approach using multiple-sequence alignment. In *Proc. of HLT/NAACL*.

Daniel Bikel, Richard Schwartz, and Ralph Weischedel. 1999. An algorithm that learns what's in a name. *Machine Learning Journal Special Issue on Natural Language Learning*, 34:211–231.

Sasha Blair-Goldensohn, Kathleen McKeown, and Andrew Hazen Schlaikjer, 2004. *Answering Definitional Questions: A Hybrid Approach*. AAAI Press.

Robin Collier. 1998. *Automatic Template Creation for Information Extraction*. Ph.D. thesis, University of Sheffield.

Pablo Duboue and Kathleen McKeown. 2003. Statistical acquisition of content selection rules for natural language generation. In *Proc. of EMNLP*.

Elena Filatova and Vasileios Hatzivassiloglou. 2003. Domain-independent detection, extraction, and labeling of atomic events. In *Proc. of RANLP*.

Elena Filatova and John Prager. 2005. Tell me what you do and I'll tell you what you are: Learning occupation-related activities for biographies. In *Proc. of EMNLP/HLT*.

Charles Fillmore and Collin Baker. 2001. Frame semantics for text understanding. In *Proc. of WordNet and Other Lexical Resources Workshop, NAACL*.

Jon Fiscus, George Doddington, John Garofolo, and Alvin Martin. 1999. NIST's 1998 topic detection and tracking evaluation (TDT2). In *Proc. of the 1999 DARPA Broadcast News Workshop*, pages 19–24.

Joseph Fleiss, Bruce Levin, and Myunghee Cho Paik, 1981. *Statistical Methods for Rates and Proportions*. J. Wiley.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288.

Sanda Harabagiu and Finley Lăcătuşu. 2005. Topic themes for multi-document summarization. In *Proc. of SIGIR*.

Sanda Harabagiu and Steven Maiorano. 2002. Multi-document summarization with GISTexter. In *Proc. of LREC*.

Jerry Hobbs and David Israel. 1994. Principles of template design. In *Proc. of the HLT Workshop*.

Silja Huttunen, Roman Yangarber, and Ralph Grishman. 2002. Complexity of event structure in IE scenarios. In *Proc. of COLING*.

Dan Klein and Christopher Manning. 2002. Fast exact inference with a factored model for natural language parsing. In *Proc. of NIPS*.

Hans Luhn. 1957. A statistical approach to mechanized encoding and searching of literary information. *IBM Journal of Research and Development*, 1:309–317.

Elaine Marsh and Dennis Perzanowski. 1997. MUC-7 evaluation of IE technology: Overview of results. In *Proc. of the 7th MUC*.

Boyan Onyshkevych. 1994. Issues and methodology for template design for information extraction system. In *Proc. of the HLT Workshop*.

Martha Palmer, Dan Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

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

Dragomir Radev and Kathleen McKeown. 1998. Generating natural language summaries from multiple on-line sources. *Computational Linguistics*, 24(3):469–500.

Ellen Riloff and Mark Schmelzenbach. 1998. An empirical approach to conceptual case frame acquisition. In *Proc. of the 6th Workshop on Very Large Corpora*.

Gerard Salton, 1971. *The SMART retrieval system*. Prentice-Hall, NJ.

Kiyoshi Sudo, Satoshi Sekine, and Ralph Grishman. 2003. An improved extraction pattern representation model for automatic IE pattern acquisition. In *Proc. of ACL*.

Ralph Weischedel, Jinxi Xu, and Ana Licuanan, 2004. *Hybrid Approach to Answering Biographical Questions*. AAAI Press.

Michael White, Tanya Korelsky, Claire Cardie, Vincent Ng, David Pierce, and Kiri Wagstaff. 2001. Multi-document summarization via information extraction. In *Proc. of HLT*.

Mohammed Zaki. 2002. Efficiently mining frequent trees in a forest. In *Proc. of SIGKDD*.

Liang Zhou, Miruna Ticrea, and Eduard Hovy. 2004. Multi-document biography summarization. In *Proc. of EMNLP*.