Automatic Creation of Domain Templates

Abstract

Recently, many Natural Language Processing applications have improved the quality of their output by using various machine learning techniques to mine Information Extraction patterns for capturing information from the input text. Currently, to mine IE patterns one should know in advance the type of the information which 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 has not been addressed before, we propose an evaluation procedure 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, some of them proscribed by the question type. A biography, for example, might contain the date of birth, the occupation, education, or the nationality of the person in question (Duboue and McKeown, 2003; Zhou et al., 2004; Weischedel et al., 2004). 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, location, victims, perpetrator and date as the templates designed for MUC (White et al., 2001) predict. Furthermore, the type of information included may vary depending on context. A biography of a movie star would include movie names, while a biography of an inventor would include the names of inventions. A description of a terrorist event in the seventies 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). This fixes the nature of the response with no flexibility depending on context. 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 a response to question requiring a description of an event. The template can be tailored to a specific time period or country simply by changing the documents from which learning takes place.

In this paper we present a method of domain-independent on-the-fly template creation. Our algorithm is completely automatic. As input it requires several collections of documents describing different instances of a domain. For example, for the *earthquake* domain, two of the instances would be: the earthquake in Japan on the 26 of October, 2004 and the earthquake in Afghanistan on the 25 of March, 2002.

We propose a robust method for domain template creation. To create a domain template we perform an annalists across different instances of this domain. We automatically identify verbs important for the domain and then learn what relations containing these verbs are typical for all the instances of the domain. To identify such relations we use shallow semantic and syntactic analysis of the document collections corresponding to each instance of the domain. These common relations are used to create the domain template. Our evaluation shows that sentences extracted according to the created templates for new domain instances contain from 50% to 100% of the important information depending on the domain. These results are promising, especially taking into consideration that we do not use any domain-specific pre-defined information.

In the remainder of this paper, we first discuss related work and then describe the process of manual knowledge acquisition by experts. We then turn to the data collections used for our experiments and the approach which we propose for automatic domain template creation. Our evaluation shows that the automatically created domain templates match human expectations about the do-

mains.

2 Related Work

In this paper we investigate the task of automatic creation of templates for capturing the most important pieces of information for a particular domain. According to (Hobbs and Israel, 1994), the problem of template creation is an instance of the problem of knowledge representation. A template is a linguistic pattern, usually a set of attribute-value pairs, with the values being text strings. The templates are normally created manually by experts to capture the structure of the facts sought in a given domain.

Research closest to automatic template creation was addressed within the Conceptual Case Frame Acquisition project (Riloff and Schmelzenbach, 1998). In this work, extraction patterns and a semantic lexicon for a domain are used to produce multi-slot case frames with selectional restrictions. The system requires two sets of documents: one set contains the documents relevant to the domain of interest and the other set contains documents which are not relevant to this domain. The system also requires as input a list of conceptual roles and associated semantic categories for the domain. A human annotator in the loop reviews and filters dictionaries of ranked extracted patterns and category words. These requirements can be fulfilled only if the reviewer who filters the dictionaries and creates the list of associated semantic categories has some knowledge of the domain under investigation. In our work we do not require any domain-dependant knowledge and use only corpus-based statistics.

The GISTEXTER summarization system (Harabagiu and Lăcătuşu, 2002) investigated the possibility of using statistics over the document collection together with semantic relations extracted from WordNet. The templates created by GISTEXTER model an input collection of documents rather then a domain and heavily depend on the topical relations encoded into WordNet. In our work, we learn templates from several collections of documents aiming for a general domain template. We rely on the relations which are cross-mentioned in different instances of the domain rather then on WordNet topic relations used in one of these instances.

To our knowledge, the only prior work on fully automatic creation of templates for IE tasks was done by Collier (1998). This method relied on Luhn's idea (Luhn, 1957) of locating statistically significant words in a corpus and used those to locate the sentences in which they occur. Then, in those sentences, Subject-Verb-Object structures were analyzed to extract the interactions mentioned in those sentences. This system was constructed to create MUC templates for *terrorist attacks*. In this work we rely on corpus statistics as well. We propose a novel approach for template creation and test it on different domains.

3 Our Approach to Template Creation

Deciding which information is important for a question about events is domain specific. A description of a terrorist attack will require different information than a description for an earthquake, for example. Approaches which encode this information *a priori* (e.g., the MUC templates) use a domain expert to specify the domain template slots. Current evaluations which are similar in spirit to MUC, for example ACE[1], also use predefined frames connecting verbs to the nouns of specific types which are linked to these verbs. But the process of domain template construction is time-consuming and labor-intensive. In our approach, we acquire important information automatically by analyzing instances from domains which are frequently discussed in the news articles.

After reading about different 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 an answer. This process can be called domain modelling. The more instances of a particular domain a person has seen, the better understanding this person 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 then be used to create a domain template. We test our system on four domains: airplane crashes, earthquakes, presidential elections, terrorist attacks.

Filatova and Prager (2005) use a similar approach to learn activities performed by people having the same occupation (e.g., artists, explorers, mathematicians). Analysis of document collections describing people of various occupations allows the system to learn activities typical for

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

people of these occupations as well as activities which can be used in any biography regardless of a person's occupation.

4 Data Description

4.1 Training Data

To create training document collections we used BBC Advanced Search[2] and submitted queries of the type: ⟨*domain title* + *country*⟩. For example, we submitted the following query to retrieve documents about presidential election in the US:

⟨presidential election +USA⟩.

To constrain the extracted documents to some point in time, we used the facility provided by the BBC Advanced Search, which allows the user to specify a publication date (or time period) of interest.

For each of the four domains of interest, we identify several instances of these domains. For example, for the *earthquake* domain we identified the following five instances:

1. In Afghanistan on March 25, 2002
2. In India on January 26, 2001
3. In Iran on December 26, 2003
4. In Japan on October 23, 2004
5. In Peru on June 23, 2001

Using the procedure described above we retrieved training document collections for each of several instances of the four domains of interest, as shown below:

- Airplane crashes (9 instances)
- Earthquakes (5 instances)
- Presidential election (13 instances)
- Terrorist attack (6 instances)

4.2 Test Data

For testing our system, we used documents from the Topic Detection and Tracking (TDT) corpus (Fiscus et al., 1999). Each TDT topic is associated with a collection and is similar to what we call an instance of a domain. TDT topics are assigned to categories, such as *Accidents* or *Natural Disasters*[3]. These categories are quite broad. From the topics corresponding to these broad categories (e.g., Accidents), we manually identified instances of the domains we chose from BBC News (e.g., Airplane crashes).

For out testing experiments we used the documents from TDT-2 and TDT-4 corpora corresponding to the following TDT topics (in brackets we specify the TDT topic IDs):

- Airplane crashes: 2 topics (20, 41026;)
- Earthquakes: 3 topics (89, 40021, 40038;)
- Presidential elections: 6 topics (40007, 40016, 41019, 41021, 41022, 41027;)
- Terrorist attacks: 3 topics (40059, 41016, 41020).

5 Creating Templates

In this work we build domain templates around verbs which are estimated to be important for the domains under investigation. 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; 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. Then, we use these corpus statistics to create domain templates from these subtrees.

For each of the four domains described in Section 4, we automatically create domain templates using the algorithm given in Section 5.1.

5.1 Algorithm for Automatic Template Creation

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

$$P(lex_i) = \frac{count_{lex_i}}{\sum_{lex_j \in comb\ coll} count_{lex_j}} \quad (1)$$

We discard those verbs that are stop words (Salton, 1971). Then, for each domain we analyze the top 50 verbs. These verbs have highest probabilities for the combined document collections for all instances in the domain and thus, we build the domain template slots around these verbs. For example, the top ten words for the *terrorist attack* domain are:

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

Step 2: Parse those sentences which contain the top 50 verbs. After we identify the 50 most frequent verbs for the domain under analysis we create syntactic parse trees for all the sentences in

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

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

elements	subtree
8	(SBAR(S(VP(VBDkilled)(NP(QP(INat))(NNSpeople)))))
8	(SBAR(S(VP(VBDkilled)(NP(QP(JJSleast))(NNSpeople)))))
5	(VP(ADVP)(VBDkilled)(NP(NNSpeople)))
6	(VP(VBDkilled)(NP(ADJP(JJmany))(NNSpeople)))
5	(VP(VP(VBDkilled)(NP(NNSpeople))))
7	(VP(ADVP(NP))(VBDkilled)(NP(CD34)(NNSpeople)))
6	(VP(ADVP)(VBDkilled)(NP(CD34)(NNSpeople)))

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

the domain document collection containing these verbs. To do this we use 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 nouns 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[4]. This software is an implementation of the algorithm presented independently by two groups (Abe et al., 2002; Zaki, 2002), which efficiently extracts frequent ordered subtrees from a set of ordered trees (forest database). Most of the subtrees which we analyze are either simple clauses or VPs. 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 elements, where an element is either a lexeme or a syntactic tag corresponding to this lexeme. 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 some location or date, 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 elements are in the corresponding subtree.

Step 4: Substitute named entities with their respective tags. We are interested in analyzing a whole domain, not the particular instance in this domain. Thus, we substitute all the named entities with their respective tags, and all the exact numbers with a tag NUMBER. This is a crucial point in our algorithm; 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. To get named entity tags for our corpus we used BBN IdentiFinder (Bikel et al., 1999). The procedure of substituting named entities with their respective tags previously proved to be useful for identification of activities common for representatives of some occupation (Filatova and Prager, 2005).

Step 5: Merge together the frequent subtrees. Finally, we merge together those subtrees which are identical according to the syntactic 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, though the subtrees themselves are not identical, we merge them into one subtree containing the longest common pattern:

(VBDkilled)(NP(CDnumber)(NNSpeople))

After this merging procedure we keep only subtrees which are used in all the instances of the domain; afterwards, we analyze only those subtrees which are used in every instance of the domain at least twice. These restrictions allow us to keep only those subtrees which potentially contain important information that should be included in the domain template. We also remove all the syntactic dependencies as we want to make this pattern as general for the domain as possible. Such a pattern without syntactic dependencies contains a verb together with a prospective template slot corresponding to this verb:

killed: (CDnumber) (NNSpeople)

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, one for the *airplane crash* domain, and the other for the *earthquake* domain, are shown below:

(ORGANIZATION) crashed
(NNearthquake) struck

We term the structure with consisting of a verb together with all the slots associated with, a *slot structure*. Here is a part of the slot structure we get for the verb *killed* after cross-examination of the *terrorist attack* domain instances:

killed (CDnumber) (NNSpeople)
(PERSON) killed
(NNsuicide) killed

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

Slot structures are similar to verb frames, which are manually created as part of the PropBank annotation (Palmer et al., 2005)[5]. An example of the PropBank frame for the verb *to kill* is:

Roleset kill.01 "cause to die":
Roles:
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 semantic type to each slot, while our algorithm gives either the type of the named entity which should fill in this slot or puts a particular noun into the slot (e.g., ORGANIZATION, earthquake, people).

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 domain important verbs are called domain templates. After we get the templates for all four domains we check for those template slots which are used in all the domains and remove them from the templates. An example of a template slot used in multiple domains is:

(PERSON) told.

□

This algorithm allows the creation of a domain template for any domain. This algorithm does not rely on any pre-defined domain or world knowledge. Our approach allows learning domain templates from cross-examination of the document collections describing different instances of the domain of interest. Next, we evaluate our approach.

6 Evaluation

The task we deal with in this paper is new and there is no well-defined and standardized evaluation procedure for this task. 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 needed to create a gold standard. In Section 6.1, we describe the procedure of how we create a gold standard.

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

Then, in Section 6.2, we evaluate the quality of the automatically created templates by extracting sentences corresponding to the templates and verifying how many answers from the questions created as gold standard are answered by the extracted sentences.

6.1 Stage 1. Information Included into Templates: Interannotator Agreement

In our first evaluation experiment, we evaluate the agreement among annotators on what information should be included into the template. For this evaluation 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 most important aspects for the domain and not how well the subjects agree on the structure of the template.

Here are the instructions we gave to our subjects to create a list of questions corresponding to the *Presidential Election* domain.

1. Imagine you are a journalist who often writes briefs on some particular domain. For example, about different cases of *Presidential Elections*. In these briefs you include the most important and at the same time general aspects about these events.

2. For this task, do not concentrate on a particular case of a presidential election (particular country or year). Rather, imagine what events are typical for a general case of a presidential election and are likely to be used in the description of any case of presidential election (e.g., presidential election in Azerbaijan in 2003, Georgia in 2004, Taiwan in 2004, US in 2004).

3. If you think that you are not very familiar with the domain you can use any resource you want to get more information about this domain (for example, search Google, or check encyclopedia articles)

4. Now, write up to 20 questions covering the most important aspects of the events which you consider the most important for this domain. Information about these you will try to find to include into your brief. For example,

	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.

for the *Summit* domain, some of the questions would be:

- Who participated in the summit?
- When did the summit happen?
- Where did the summit happen?
- What subject was discussed?
- Whether any document was signed or not?
- Etc.

5. Do not try to write exactly 20 questions. If you believe that fewer questions are enough for a complete description of the term, then stop there.

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

Usually, the degree of interannotator agreement is estimated using Kappa. For this task, though, Kappa statistics cannot be used as it requires knowledge of the expected or chance agreement, which is not applicable to this task (Fleiss et al., 1981). To measure interannotator agreement we use 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 according to the following formula:

$$Jaccard(domain_d) = \frac{|QSd_i \cap QSd_j|}{|QSd_i \cup QSd_j|} \quad (2)$$

where QSd_i and QSd_j 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 sum of the pair wise 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. For example, one of our subjects does not even ask about the outcome of the election. 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 presented in Table 2

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

6.2 Stage 2. Quality of the Automatically Created Templates

In section 6.1 we demonstrate 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 templates created automatically for four domains.

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 sentences extracted according to the created templates and

check whether these sentences contain the answers to the questions created by the subjects during the first stage of the evaluation.

We evaluate sentences extracted for every test instance of all the domains. We extract the sentences corresponding to the test instances according to the following procedure:

1. Break all the documents corresponding to a particular test instance (respective TDT topic) into sentences.
2. Analyze the template created for the respective domain. For every domain template slot analyze all the sentences corresponding to the instance (TDT topic) under analysis. Find the first sentence which contains this slot. Add this sentence to the list of extracted sentences unless, this sentence has been already added to this list.
3. Keep adding sentences to the list of extracted sentences till all the template slots are analyzed or the size of the list exceeds 20 sentences.

The key step in the above algorithm is Step 2. By extracting only one sentence corresponding to a particular slot we prevent the output for evaluation to grow enormously, as the sentences corresponding to one slot often exceeds 100. There are various techniques to choose one sentence from a large set of sentences. We deliberately do not try to optimize this choice as we do not want to give any advantage to our system.

In Step 3 we keep only first twenty sentences so that the number of sentences which potentially contain 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 probability of the verb for the domain the closer to the top of the template the slot structure corresponding to this verb. We assume that the important information is more likely to be covered by the slot structures which are placed on the top of the template.

The evaluation results for the automatically created templates are presented in Figure 1. This figure shows the number of questions for which answers are captured by the extracted sentences. For each subject within each domain we calculated the average ratio across all the test domain instances. We did the same for the intersection. One can notice that the best results we got are for the *airplane crash* domain, and the worst results are for the *presidential election* domain. Poor results for the *presidential election* domain could be predicted from the Jaccard metric value for interannotator agreement(Table 2). 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 frauds or about opposition taking over the power without the formal resignation of the previous president. For such TDT topics the number of questions answered by the extracted answers is the lowest.

We also did not extract answers for those questions which require an explanation or a definition answer. For example, one of the questions asked for the *earthquake* domain is:

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

This question corresponds to the following three questions created by the other subject for this domain:

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

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

We would like to note the number of answers covered for the questions from the intersection of question lists is the highest number for all four domains. Intersection of the question lists contains the questions which target information considered important by all the subjects who created questions for the domain.

Overall, this evaluation shows that using automatically created domain templates we extract sentences which contain at least 50% of the important information, as in the case of *presidential election* domain. For those domains which have small diversity this number can be as high as 100%, as in the case of *airplane crash* domain.

7 Conclusions

In this paper, we have 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

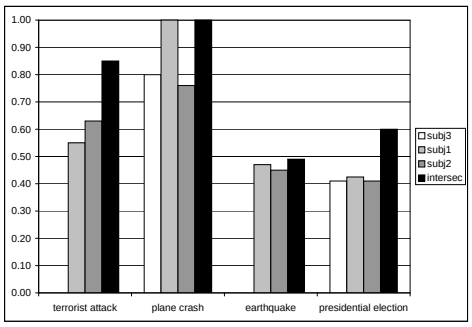

Figure 1: Evaluation results.

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; through application to document collections in different time periods or different locales, we can learn domain descriptions that are tailored to time period or location.

References

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

Amit Bagga. 1997. Analyzing the complexity of a domain with respect to an Information Extraction task. In *Proceedings of the Seventh MUC Conference*.

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, Department of Computer Science.

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

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 *Proceedings of the EMNLP/HLT Conference*.

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

Jon Fiscus, George Doddington, John Garofolo, and Alvin Martin. 1999. NIST's 1998 topic detectoin and tracking evaluation (TDT2). In *Proceedings of the 1999 DARPA Broadcast News Workshop*.

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

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

Sanda Harabagiu and Finley Lăcătuşu. 2002. Generating single and multi-document summaries with GISTexter. In *Proceedings of the DUC Conference*.

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

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

Dan Klein and Christopher Manning. 2002. Fast exact inference with a factored model for natural language parsing. In *Proceedings of Advances in Neural Information Processing Systems 15 (NIPS)*.

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

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 *Proceedings of the HLT/NAACL Conference*.

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

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

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 *Proceedings of the HLT Conference*.

Mohammed Zaki. 2002. Efficiently mining frequent trees in a forest. In *Proceedings of the ACM SIGKDD Conference*.

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