Combining Relations for Information Extraction from Free Text

MSTISLAV MASLENNIKOV and TAT-SENG CHUA
National University of Singapore

Relations between entities of the same semantic type tend to be sparse in free texts. Therefore, combining relations is the key to effective information extraction (IE) on free text datasets with a small set of training samples. Previous approaches to bootstrapping for IE used different types of relations, such as dependency or co-occurrence, and faced the problems of paraphrasing and misalignment of instances. To cope with these problems, we propose a framework that integrates several types of relations. After extracting candidate entities, our framework evaluates relations between them at the phrasal, dependency, semantic frame, and discourse levels. For each of these levels, we build a classifier that outputs a score for relation instances. In order to integrate these scores, we propose three strategies: (1) integrate evaluation scores from each relation classifier; (2) incorporate the elimination of negatively labeled instances in a previous strategy; and (3) add cascading of extracted relations into strategy (2). Our framework improves the state-of-art results for supervised systems by 8%, 15%, 3%, and 5% on MUC4 (terrorism); MUC6 (management succession); ACE RDC 2003 (news, general types); and ACE RDC 2003 (news, specific types) domains respectively.

Categories and Subject Descriptors: H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing—*Linguistic processing*; I.2.7 [**Artificial Intelligence**]: Natural Language Processing—*Text analysis*

General Terms: Algorithms, Experimentation, Performance

Additional Key Words and Phrases: Information extraction, dependency relations, semantic relations, discourse relations, bootstrapping

ACM Reference Format:
Maslennikov, M. and Chua, T.-S. 2010. Combining relations for information extraction from free text. ACM Trans. Inform. Syst. 28, 3, Article 14 (June 2010), 35 pages.
DOI = 10.1145/1777432.1777437 http://doi.acm.org/10.1145/1777432.1777437

1. INTRODUCTION

This article tackles the relation-related information extraction (IE) from free text. We consider several variations of this task, such as extracting

Authors' address: Department of Computer Science, School of Computing, National University of Singapore, Singapore, 117543; email: steve@oit.cmc.msu.ru.

DOI 10.1145/1777432.1777437 http://doi.acm.org/10.1145/1777432.1777437

(a) template slots (slots *Perpetrator, Victim, Target* in the terrorism domain, Message Understanding Conference 4 (MUC4)); (b) template events (succession templates <*Organization, PersonIn, PersonOut, Post*> in the management succession domain, Message Understanding Conference 6 (MUC6)); and (c) relations (Automatic Content Extraction 2003 evaluation, Relation Detection and Characterization (ACE RDC 2003) task). In all these variations, it is important to evaluate relations between entities in order to extract the correct slots in a template. We tackle this problem in our proposed approach, called ARE (Anchors and RElations), which extracts key phrases (anchors) from sentences and evaluates possible relations between anchors. Based on this evaluation, ARE fills up slots and extracts relevant templates.

In our work, we address the problem of data sparseness of the training corpus. Several previous studies attempted to tackle this problem using either bootstrapping, cotraining, counter-training or cascading paradigms. As an example, Niu et al. [2003] cascaded dependency-based rules into the hidden Markov model classifier. However, their method was applied to the named entity (NE) extraction task. The majority of relations in this task are short-distance dependency relations, which can be handled effectively by current dependency (Minipar by Lin [1997]), and ASSERT [Pradhan et al. 2004] parsers. This is in contrast to event-based IE tasks such as MUC4, in which the extracted slots may be distributed across several clauses or sentences. This causes problems in the paraphrasing and alignment of tokens. These problems may occur at different levels of granularity: phrasal, syntactic, semantic, or discourse.

Hence we have to tackle the problem of text paraphrasing, which is caused by large variations, of text representation, and leads to a significant decrease in training performance because systems are not able to unify training instances effectively. As an example, consider two instances "company named John *as* chairman" and "chairman John *was* named *by* company"; these paraphrased instances describe the post 'chairman'. The context tokens in these sentences are located in different positions and opposite directions. The state-of-the-art contextual models such as Riloff et al. [2005], Chieu et al. [2002], and Xiao et al. [2004] that extract generalized patterns based on the contextual token positions of one sentence structure will be unable to handle the paraphrased structure of the other. However, if we examine the entity dependency relations of these two paraphrased sentences, we will find that they are actually invariant with: *"subj:company verb:named person:John as obj:president"*. Thus, it is important to utilize meaningful entities and relations when performing IE, rather than only relying on context tokens.

Another important problem in IE systems is the alignment of tokens. Insertion or deletion of several tokens lead to different positions of other tokens, which may result in data sparseness. For example, consider the two instances "X *was* killed *by* terrorists" and "X *was* killed *this week by* terrorists". The extra tokens "this week" in the second instance lead to misalignment of the tokens "by terrorists". Therefore, the context-based approaches that rely on actual token positions will lead to sparseness of data. Soft patterns by Xiao et al. [2004] attempt to alleviate this problem by performing a probabilistic

fuzzy matching of contextual tokens. However, fuzzy matching is useful only if the variations in token positions are small (1–2 tokens). A longer insertion such as "X *was* killed *during his travel on this week by* terrorists" will result in a mistake for the soft patterns too. Therefore, a reliable solution of the alignment problem has to utilize both important words or anchors and their relations.

Previous approaches to paraphrasing and alignment problems relied on co-occurrence [Xiao et al. 2004] and dependency [Zhang et al. 2006] and semantic [Surdeanu et al. 2003] relations between entities. These relations enable us to make reliable extraction of correct entities/relations at the level of a single clause. However, Maslennikov et al. [2006] reported that the increase of relation path length will lead to considerable decrease in performance. This decrease in performance occurs because entities may belong to textual parts of different granularity levels. If entities belong to different clauses, then most parsers (e. g., dependency parser Minipar by Lin [1997]) tend to fail due to long distances between entities. Since clauses in a sentence are connected by clausal relations [Halliday and Hasan 1976], thus it is important to perform discourse analysis of a sentence. Additionally, semantic analysis helps to connect entities that belong to the same predicate.

Discourse analysis may contribute to IE in several ways. First, Taboada and Mann [2005] reported that discourse analysis helps to decompose long sentences into clauses. Therefore, it helps to distinguish relevant clauses from nonrelevant ones. Second, Miltsakaki [2003] stated that entities in subordinate clauses are less salient. Third, the knowledge of textual structure helps to interpret the meaning of entities in a text [Grosz and Sidner 1986]. As an example, consider the sentences *"ABC Co. appointed a new chairman. Additionally, the current CEO was retired"*. The word 'additionally' connects the event in the second sentence to the entity 'ABC Co.' in the first sentence. Fourth, Moens and De Busser [2002] reported that discourse segments tend to be in a fixed order for structured texts such as court decisions or news. Hence, analysis of discourse order may reduce the variability of possible relations between entities.

To model these factors, we propose a multiresolution framework ARE that integrates discourse, semantic, and dependency relations at two stages. The first stage extracts candidate noun/verb phrases (called anchors) that are likely to contain an answer. The second stage evaluates the relations between these anchors. ARE aims to filter noisy dependency relations from training and support their evaluation with discourse relations between entities. Additionally, we encode semantic roles of entities in order to utilize semantic relations. Evaluations on MUC4, MUC6, and ACE RDC 2003 corpora demonstrate that our approach outperforms the state-of-art systems, due mainly to the modeling of multiple relations between candidate entities.

Additionally, we need to address two important problems in order to improve performance. First, the number of negative relation paths significantly exceeds the number of positive relation paths. We tackled this problem by training the negative relation classifier. Second, Maslennikov et al. [2006] reported that the performance of the dependency classifier on long dependency paths is

significantly lower than that on short dependency paths. Therefore, it is important to distinguish long and short positive relation paths. For this purpose, we cascaded the discourse classifier into the dependency classifier.

The contribution of this article is in combining discourse, dependency and semantic relations to tackle the data sparseness problem in IE. The framework enables us to connect entities in different clauses, and thus improve the performance of long-distance dependency paths. In addition to our previous work [Maslennikov et al. 2006; Maslennikov and Chua 2007], we filtered the negative relation paths using the negative classifier and cascaded the discourse classifier into the dependency classifier in order to separate long and short dependency paths. To test whether the data sparseness problem is alleviated with our approach, we conducted the experiments on the original training data from the MUC4, MUC6, and ACE RDC 2003 domains.

The remaining parts of the article are organized as follows. Section 2 discusses related work, and Section 3 highlights our motivation for using relations. Section 4 introduces our approach for constructing ARE, while Section 5 presents several strategies to combine different relation types and construct a template. Section 6 describes our experimental setups and results and, finally, Section 7 concludes the article.

2. RELATED WORK

Numerous attempts to tackle the data sparseness problem were conducted based on the idea of bootstrapping [Efron 1979; Brin 1998] bootstrapped the regular expression patterns for book name extraction based on the html and url structures. On the newspaper articles domain, Agichtein and Gravano [2000] were constructing LOC-ORG templates based on the probabilities of words between LOC-ORG and in the left and right contexts. They argued that this method produces selective patterns with high coverage (P = 85%, R = 82% on the corpus of WSJ, NYT news articles). Yangarber et al. [2002] extended the list of accepted names based on noun groups (e.g., adding "yellow fever vaccine" once "yellow fever" is in the list) and reported P = 50%, R = 70% for the recognition of diseases on the ProMed domain. Thelen and Riloff [2002] discovered that precision tends to be higher when a category classifier cannot learn a name that is already learned. Starting from additional ten most frequent noun phrases as negative classes, Lin et al. [2003] improved the precision to P = 70% at R = 70% for the recognition of diseases on the ProMed domain. However, these approaches use surface patterns, which may be insufficient if the distance between entities is big. Thus, it may cause a problem when instances are paraphrased at different levels: syntactic, semantic, and discourse.

Another direction of recent studies explores the idea of utilizing several learners to train each other. Although this idea is fairly intuitive, related studies reported a lot of problems. First, Riloff and Jones [1999] reported that errors in extracted entities (they extracted names) can rapidly propagate. Hence they validated the extracted entities using different learners. Second, the stopping

criteria for each learner has to depend on other learners. For example, in the scenarios of Management Succession and Natural Disasters, Yangarber [2003] found that two learners start to depend on each other when they extract the same instances such as the "Person died". Thus one possible approach is to stop the training when the extractions from two learners on different scenarios start to intersect. This has resulted in improving the performance of P = 83%, R = 83% in extracting a management succession scenario in the Message Understanding Conference 6 (MUC6). However, Yangarber also reported that this method leads to problems of ambiguity both at the document and pattern levels. Third, learners may differ significantly in their precision and recall. To overcome this problem, several authors used the cascading strategy. In the named entity extraction task, Niu et al. [2003] cascaded the high-precision rule output into the HMM classifier. Their cascading strategy allowed them to achieve state-of-art performance on the person and organization types for the MUC7 named entity recognition task. Xiao et al. [2004] cascaded the output of soft matching rules [Cui et al. 2004] into the hard matching rules of Xiao et al. [2003] and achieved $F_1 = 55\%$ based on 20% of training corpora. This is only 2% less than the results of GRID based on 100% of training corpora ($F_1 = 57\%$). Whereas these approaches are useful to integrate different levels of text, they lack a systematic perspective on different relation levels. The word alignment issue is serious for IE due to the paraphrasing, addition/deletion of noisy tokens as adjectives or punctuation. According to Xiao et al. [2004] systems that utilize hard matching rules on free text lose in recall to systems that employed statistical models. Additionally, even if two learners are able to train each other, neither learner will deal with entities in different clauses/semantic frames in these systems.

Several researchers [Yangarber 2003; Niu et al. 2003] discovered that the choice of seeds for bootstrapping is crucial to better effectiveness. Moreover, Liu et al. [2003] argued that the lack of negative seeds leads to a strong bias towards positive instances. For the problem of relation extraction, Agichtein [2003] corrected this bias by (a) splitting positive seeds into two parts, $Part_1$ and $Part_2$; and (b) using $Part_2$ to control the learning from $Part_1$. In our opinion, the success of this method largely depends on the chosen linguistic features of instances. Etzioni et al. [2005] used automatic identification of subclasses to eliminate the problem of seed selection. They performed automatic gathering of seeds using general-level heuristics (e.g., "<class1> such as NPList"). This approach is promising, and brings in the problem of automatic NE class extraction. Nevertheless, it relies on predefined surface patterns only, which misses instances that may not be covered by such heuristics. On the other hand, Davidov et al. [2007] suggested that it is sufficient to check that two words are associated with some cue, as in the form "Word1 Cue Word2". Indeed, even two seeds of the same class are sufficient to bootstrap the whole semantic class in their approach. However, these approaches tend to avoid the situation where entities belong to different clauses or semantic frames.

Recent work in IE focuses on relation, semantic, parsing and discourse-based approaches. Several recent research efforts were based on modeling relations between entities. Roth and Yih [2002] employed the Bayesian inference

model for relationship recognition. They argued that named entities (NE) and relationships support each other. Thus, learning has to be done together in order to avoid the propagation of mistakes in NE-relationship extraction cycles. Nevertheless, it is unclear wether their idea is extendable to the IE task. Bunescu and Mooney [2004] modeled relationships using undirected graphs, namely, using the relational Markov models approach. They reported a slight increase in performance for the biomedical domain in comparison to the use of conditional random fields (CRF). Culotta and Sorensen [2004] extracted relationships using dependency-based kernel trees in support vector machines (SVM). They achieved an F_1-measure of 63% in relation detection, and reported that the primary source of mistakes comes from the heterogeneous nature of nonrelation instances. Consequently, to overcome such mistakes, we need to perform additional analysis of relations at different levels. To facilitate the elimination of the nonrelation instances, we analyze them according to their discourse, semantic, and dependency levels in the proposed approach of ARE.

Another possible direction to tackle this problem is to carry out further relationship classification, similar to that of Culotta and Sorensen [2004]. Cui et al. [2005] performed flexible matching between relations to reduce the problem of heterogeneity in sentences. However, they reported that flexible matching of relations is still ineffective for cases of paraphrasing that arise from long distance relations. Maslennikov et al. [2006] classified the relation path between candidate entities into simple, average, and hard cases. This classification is based on the length of the connecting path in the dependency parse tree. The authors reported that dependency relations are not reliable for the hard cases, which may need the analysis of discourse relations to supplement dependency relation paths.

Surdeanu et al. [2003] applied semantic parsing to capture the predicate-argument sentence structure. They suggested that semantic parsing is useful in capturing verb arguments which may be connected by long-distance dependency paths. However, current semantic parsers such as the ASSERT are not able to recognize support verb constructions such as *"X conducted an attack on Y"* under the verb frame *"attack"* [Pradhan et al. 2004]. Hence, many useful predicate-argument structures will be missed. Moreover, semantic parsing belongs to the intra-clausal level of sentence analysis, which, as in the dependency case, will need the support of discourse analysis to bridge inter-clausal relations.

Webber et al. [2002] reported that discourse structure helps to extract anaphoric relations. However, their set of grammatical rules is heuristic. For the purpose of scalability, however, we need to develop an automated approach that is portable across several domains. Cimiano et al. [2005] also employed a discourse-based analysis for IE. However, their approach requires a predefined domain-dependent ontology in the format of an extended logical description grammar as described by Cimiano and Reely [2003]. Moreover, they used discourse relations between events, whereas in our approach discourse relations connect entities.

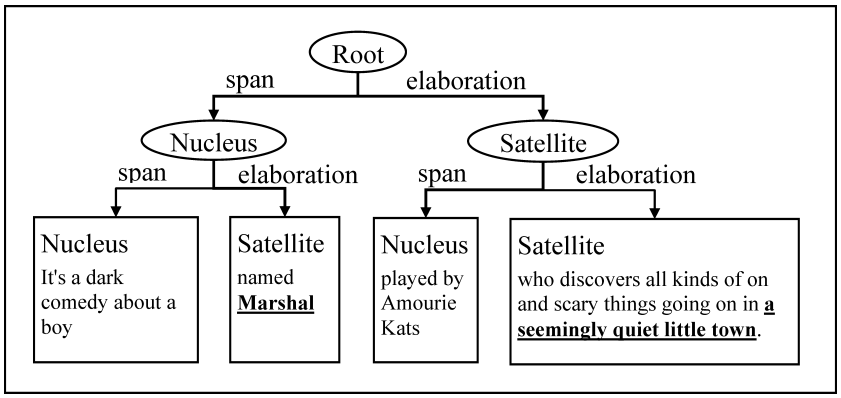

Fig. 1. Example of discourse parsing; the target entities and discourse relations between them are highlighted in bold font.

3. MOTIVATION FOR USING RELATIONS

In this section, we describe our motivation for integrating discourse, semantic, and dependency relations for IE. Our description emphasizes discourse relations because they belong to the interclausal level.

3.1 Discourse Relations

Our discourse analysis is based on the rhetorical structure theory (RST) by Taboada and Mann [2005]. RST splits the texts into two parts: (a) nuclei, the most important parts of texts; and (b) satellites, the secondary parts. We can often remove satellites without losing the meaning of the text. Both nuclei and satellites are connected with discourse relations in a hierarchical structure. In our work, we use 16 classes of discourse relations between clauses, as described in Carlson et al. [2001]. They are: *Attribution, Background, Cause, Comparison, Condition, Contrast, Elaboration, Enablement, Evaluation, Explanation, Joint, Manner-Means, Topic-Comment, Summary, Temporal, Topic-Change*. The additional three relations impose a tree structure: *textual-organization, span* and *same-unit*. All the discourse relation classes are potentially useful, since they encode some knowledge about the textual structure. Therefore, we decided to include all of them in the learning process to learn patterns with the best possible performance.

We consider two main rationales for utilizing discourse relations to IE. First, discourse relations help to narrow the search space to the level of a single clause. For example, the sentence *"[<Soc-A1> **Trudeau**</>'s <Soc-A2> **son**</> told everyone], [their prime minister was his father], [who took him to a secret base in the arctic] [and let him peek through a window]."* contains 4 clauses and 7 anchor cues (key phrases) for the type Social, which leads to 21 possible variants. Splitting this sentence into clauses reduces the combinations to 4 possible variants. Additionally, this reduction eliminates the long and noisy dependency paths.

Second, discourse analysis enables us to connect entities in different clauses with clausal relations. As an example, we consider a sentence *"It's a dark*

comedy about a boy named <AT-A1>Marshal</> played by Amourie Kats who discovers all kinds of on and scary things going on in <AT-A2>a seemingly quiet little town</>". In this example, we need to extract the relation *"At"* between the entities *"Marshal"* and *"a seemingly quiet little town"*. The discourse structure of this sentence is given in Figure 1. The discourse path *"Marshal <-elaboration- _ <-span- _ -elaboration-> _ -elaboration-> town"* is relatively short and captures the necessary relations. At the same time, prediction based on dependency path *"Marshal <–obj- _ <-i- _ <-fc- _ <-pnmod- _ <-pred- _ <-i- _ <-null- _ -null-> _ -rel-> _ -i-> _ -mod-> _ -pcomp-n-> town"* is unreliable, since the relation path is long. Thus, it is important to rely on discourse analysis in this example. In addition, we need to evaluate both the score and reliability of prediction by relation path of each type.

3.2 Dependency and Semantic Relations

We use dependency relations because they capture paraphrasing at the syntactic level. This is important to unify instances such as "terrorists attacked peasants", "peasants *were* attacked *by* terrorists" and "peasants, *working in a field*, *were* attacked *by* terrorists", which have the same dependency structure. Therefore, the subject-verb-object relations between anchors, 'terrorists', 'attacked' and 'peasants,' are crucial for the learning model.

We also need semantic relations because they connect distant phrases that belong to the same predicate-argument sentence structure. In the example sentence *"[ARG0 He] has written to President Vladimir Putin to [TARGET request] [ARG1 his release]"*, the predicate 'request' is connected with the argument 'he'.

The discourse, dependency and semantic relations between anchors given above are based on parser output in the hierarchical format. Therefore, it is possible to process these relations in a similar manner. In this way, our task is reduced to separately evaluating different relation types and subsequently combining them.

4. ANCHOR AND RELATION MODEL

In contrast to current context-based IE approaches, the aim of ARE is to tackle possible paraphrasing and alignment problems by constructing an invariant model. Intuitively, there are two primary invariant structures within instances: entities and the relations between them. So our aim is to extract relevant entities and use different kinds of relations to choose the best among them for slot fillings. We refer to such entities as anchors. As an example, consider the excerpts "Terrorists attacked victims", "Peasants *were* murdered *by unidentified* individuals" and "Soldiers *participated in* massacre *of* priests". The instances for the extraction of the VICTIM type may be represented as follows:

$\mathbf{W}_{-3}$	$\mathbf{W}_{-2}$	$\mathbf{W}_{-1}$	$\mathbf{W}_0$	$\mathbf{W}_1$	$\mathbf{W}_2$	$\mathbf{W}_3$
	terrorists	attacked	victims			
			peasants	were	murdered	by
in	massacre	of	priests			

Table I. Linguistic Features for Anchor Extraction

Anchor types / Features	Perpetrator_Cue (A)	Action_Cue (D)	Victim_Cue (A)	Target_Cue (A)
Lexical (Head noun)	terrorists, individuals, Soldiers	attacked, murder, massacre	Mayor, general, priests	bridge, house, Ministry
Part-of-Speech	Noun	Verb	Noun	Noun
Named Entities	Soldiers (PERSON)	-	Jesuit priests (PERSON)	WTC (OBJECT)
Synonyms	Synset 130, 166	Synset 22	Synset 68	Synset 71
Concept Class	ID 2, 3	ID 9	ID 22, 43	ID 61, 48
Co-referenced entity	He -> terrorist, soldier	-	They -> peasants	-
Clausal type	Nucleus, Satellite	Nucleus, Satellite	Nucleus, Satellite	Nucleus, Satellite
Argument type	Arg0	Target	Arg1	Arg2

Anchors in this example consist of: (1) words 'terrorists' as Perpetrator_Cue; (2) words 'attacked,' 'murdered,' and 'massacre' as Action_Cue; and (3) words 'victims,' 'peasants,' and 'priests' as Victim_Cue. While the invariant dependency relations are subject, verb, and object.

The next three sections describe the overall architecture of ARE, as well as our approaches for anchor cue mining, and extraction and ranking of dependency relation paths between them.

4.1 Anchor Cue Mining

In the first stage, we need to mine anchors from tokens in input sentences. Anchor cues provide a strong prediction for the presence of specific targets or event occurrences.

Every token in ARE has different levels of representation, which include Lexical, Part-of-Speech, Named Entities, Synonyms and Concept classes. The synonym set and concept classes are mainly obtained from Wordnet, while part-of-speech is obtained using the NLProcessor from Infogistics Ltd,[1] and named entities are extracted using our rule-based named entity recognizer, which is briefly introduced in Yang et al. [2003].

We classify anchor cues into two types: the general type and an action type. First, the general type corresponds to the type of slots that should be filled in a template. The terrorism domain has three general types of: Perpetrator, Victim, and Target; while the management succession domain has the general types of: Post, Person In, Person Out, and Organization. Second, the action type is used for mining cues related to a meaningful action. The terrorism action type consists of words such as 'murder' or 'attack'. In management succession we define the types Action-In (words 'hire,' 'assume') and Action-Out (words 'retire,' 'leave'). The examples of possible anchor cues (or slots) and their representations in the terrorism domain are given in Table I. The list of tokens for each anchor cue type is mined automatically from the training examples.

[1]Available at http://www.infogistics. com/textanalysis.html.

Given the set of tokens for a particular anchor cue and an input phrase, we need to classify whether the phrase belongs to this anchor cue type. Let P be an input phrase and C_j be the anchor cue we want to match. The similarity score of P for a given anchor cue C_j is given by

$$\begin{aligned} Entity\_Score(P, C_j) = {} & \delta_1 * S\_lexical(P, C_j) + \delta_2 * S\_POS(P, C_j) + \\ & \delta_3 * S\_NE(P, C_j) + \delta_4 * S\_Syn(P, C_j) + \\ & \delta_5 * S\_ConceptClass(P, C_j) \end{aligned} \quad (1)$$

where $S\_X(P,C_j)$ is a score function for a particular representation type denoted by $X = \{lexical, POS, NE, Syn, ConceptClass\}$, and δ_i is the corresponding weight for the representation type. The weights are learned automatically using expectation maximization (EM) by Dempster et al. (1977). Our main motivation for using EM is that it maximizes the expected probability of matching features.

Let us have M positive/negative examples of anchor phrases P_i in training data. Let the phrase P_i have linguistic features $f_{i1}, \ldots, f_{iN}$, where each f_{ih} is taken from Table I, and N stands for the number of linguistic features. Under the EM methodology, we assume that the membership of phrase P_j to class C_j can be modeled as a mixture of Gaussian distributions of features f_{ih}. First, we choose a set of initial weights $\delta_1(step_0) = \cdots = \delta_N(step_0)$. Next, we perform several iterations of expectation and maximization steps. At the expectation step (E-step), we calculate the expectation of contribution of each feature estimation y_{ih} to its class C_j using the current δ_i:

$$y_{ih}(step_k) = \frac{\delta_j(step_{k-1}) * P(C_j | f_{ih})}{P(C_j)} \quad (2)$$

Here $P(C_j | f_{ih})$ are the probabilities that an instance with a feature f_{ih} will belong to class C_j, and $P(C_j)$ is a probability of the class C_j. At the following maximization step (M-step), we maximize the expected probability to match anchors using the estimations $y_{ih}(step_k)$ of features. Our assumption is that the missing data are known.

$$\begin{cases} \delta_1(step_k) = \dfrac{(y_{11}(step_k) + \cdots + y_{M1}(step_k))}{M}) \\ \quad \cdots \\ \delta_N(step_k) = \frac{(y_{1N}(step_k) + \cdots + y_{MN}(step_k))}{M} \end{cases} \quad (3)$$

We iterate the EM procedure using Eqs. (2) and (3) until $\| \delta_i(step_k) - \delta_i(step_{k-1}) \| < \delta_{EM}$, where δ_{EM} is a predefined threshold.

Finally, we obtain the $Entity\_Score(P,C_j)$ of incoming phrases in Eq. (1). For this purpose, we plug-in the weights δ_i from Eq. (3). In order to extract subscores, we analyze the linguistic features of entities from slots in the training instances. Let $S\_X(P, C_j)$ correspond to the feature f_{ih} in Table I. We calculate $S\_X(P,C_j)$ as a probability that an entity with the feature f_{ih} belongs to a slot of type C_j:

$$S\_X(P, C_j) = P(C_j | f_{ih}) \quad (4)$$

Table II. Instances with Anchor Cues (Our aim is to extract victims at the position W_0)

W_{-3}	W_{-2}	W_{-1}	W_0	W_1	W_2	W_3
	Perp Cue	Action Cue	Victim Cue			
			Victim Cue	were	Action Cue	by
In	Action Cue	Of	Victim Cue			

We classify the phrase P as belonging to an anchor cue when its $Entity\_score(P,C_j)$ is above an empirically determined threshold ω.

After mining the anchors, we can extract meaningful anchor cues. Example instances with anchor cues are shown in Table II.

4.2 Relations Between Anchors

To resolve the correct filling of phrase P of type C_i in a desired slot in the template, we need to consider the relations among multiple candidate phrases of related slots. To do so, we consider several types of relations among anchors: discourse, dependency, and semantic relations. These relations capture the interactions between anchors, and hence are useful for tackling the paraphrasing and alignment problems [Maslennikov et al. 2006]. Given two anchors A_i and A_j of anchor types C_i and C_j, we consider a relation $Path_l = [A_i, Rel_1, \ldots, Rel_n, A_j]$ between them such that there are no anchors between A_i and A_j.

Additionally, we assume that the relations between anchors are represented in the form of a tree T_l, where $l = \{s, c, d\}$ refers to discourse, dependency, and semantic relation types, respectively. We describe the nodes and edges of T_l separately for each type because their representations are different:

—The nodes of a discourse tree T_c consist of clauses $[Clause_1, \ldots, Clause_{Ncl}]$; and their relation edges are obtained from the Spade system described in Soricut and Marcu [2003]. This system performs RST-based parsing at the sentence level. The reported accuracy of Spade is 49% on the RST-DT corpus. To obtain a clausal path, we map each anchor A_i to its clause in Spade. If anchors A_i and A_j belong to the same clause, we assign them the relation *same-clause*.

—The nodes of a dependency tree T_d consist of words in sentences; and their relation edges are obtained from Minipar by Lin [1997], who reported a parsing performance of precision = 88.5% and recall = 78.6% on the SUSANNE corpus.

—The nodes of a semantic tree T_s consist of arguments $[Arg_0, \ldots, Arg_{Narg}]$ and targets $[Target_1, \ldots, Target_{Ntarg}]$. Both arguments and targets are obtained from the ASSERT parser developed by Pradhan et al. [2004]. The reported performance of ASSERT is $F_1 = 83.8\%$ on the identification and classification task for all arguments, evaluated using PropBank and AQUAINT as the training and testing corpora, respectively. Since the relation edges have the form $Target_k$ -> Arg_l, the relation path in the semantic frame contains only a single relation. Therefore, we encode semantic relations as part of the anchor features.

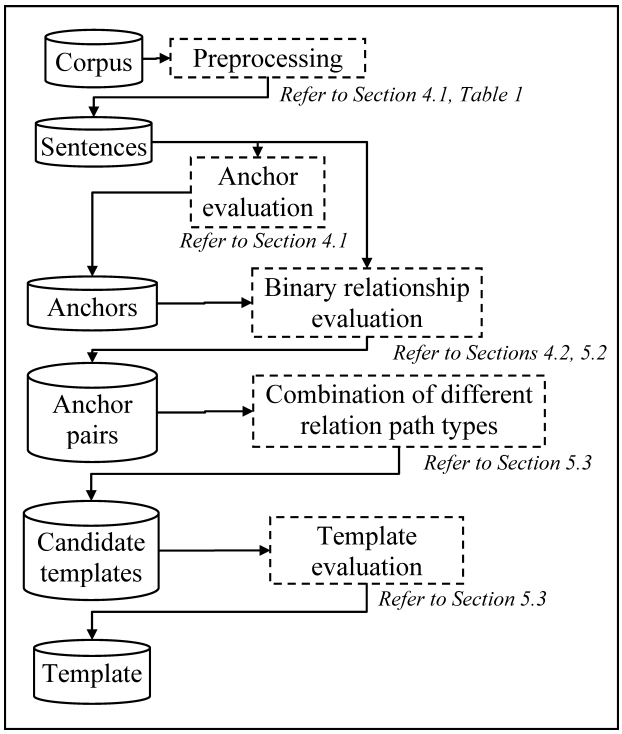

Fig. 2. Architecture of the system.

In the later parts of this article, we consider only discourse and dependency relation paths $Path_l$, where $l = \{c, d\}$.

4.3 Anchor and Relation System Architecture

For the purpose of tackling paraphrasing and alignment problems, it is important to extract candidate anchors and evaluate relations between these entities. While the extraction of anchors allows us to decrease the problem dimensionality, relations help to choose the correct anchors for filling the templates. Additionally, the same entity may participate in different templates. For example, the entity X in the sentence *"He succeeds X who was recently made chairman of Y"* belongs to two templates *"Action:succeeds PersonOut:X" and "Company:Y Action:made Post:chairman PersonIn:X"*. This example illustrates, that a single entity X may need to be assigned with two anchors of different types of PersonIn and PersonOut, which may belong to their respective templates. Whether an extracted anchor should belong to a template depends heavily on the evaluation of relations between the anchors. Therefore, our Anchor and RElation (ARE) architecture consists of the following components: preprocessing sentences, extracting candidate entities (anchors), evaluating relations between them, combinating anchor pairs, and evaluating possible templates. For the case of the relation extraction task, the final template is the same as an extracted binary relation. The architecture of our system is shown in Figure 2.

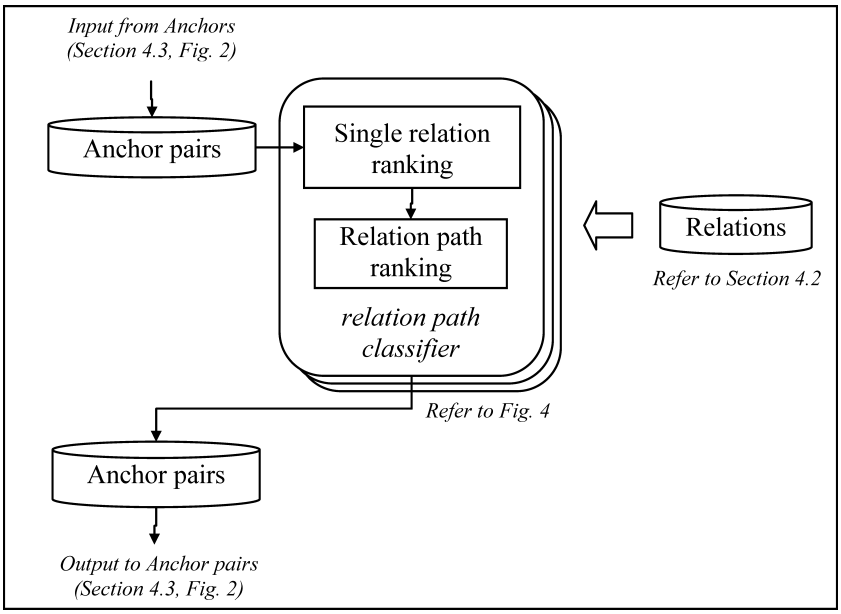

Fig. 3. Binary relationship evaluation.

In order to extract anchors, we preprocess a corpus to split sentences, extract linguistic features and parse text with Minipar, Spade, and ASSERT. This linguistic information is input into the anchor extraction module. After extracting the anchors, we evaluate the semantic, dependency, and discourse relations among them. While semantic relations are encoded in the form of linguistic features, dependency and discourse relations are represented in the form of tree paths between anchors. These path evaluations are unified into an integral path score. Finally, we fill templates based on a predefined set of slots, such as the Organization, PersonIn, PersonOut and Post for the management succession domain. Both anchors and relations are used for filling candidate templates and their evaluation.

5. OUR APPROACH

In this section we describe our relation-based approach to IE. We start with the evaluation of relation paths (single relation ranking and relation path ranking) to assess the suitability of their anchors as entities to template slots. Here, given a single relation or relation path, we want to evaluate whether the two anchors are correct in filling the appropriate slots in a template. This is followed by the integration of relation paths and the evaluation of templates.

5.1 Evaluation of Relation Path

In the first stage, we evaluate from training data the relevance of relation path, $Path_l = [A_i, Rel_1, \ldots, Rel_n, A_j]$ between the candidate anchors A_i and A_j of types C_i and C_j. We divide this task into two steps, as given in Figure 3. The first step ranks each single relation $Rel_k \in Path_l$; while the second step combines the evaluations of Rel_k to rank the whole relation path $Path_l$.

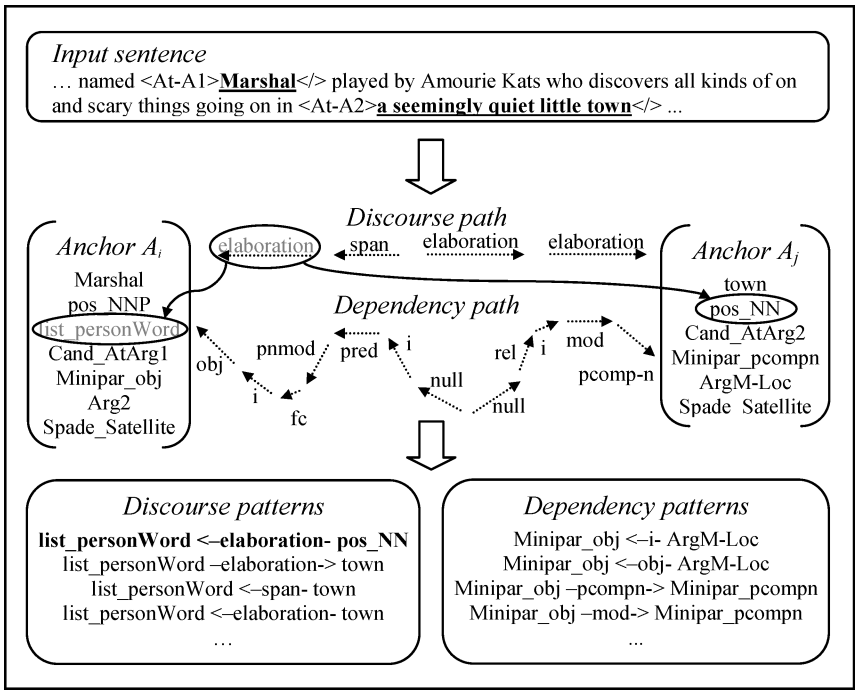

Fig. 4. Examples of discourse and dependency patterns. We use a linguistic feature from both anchors and a relation; a sample result pattern is highlighted in bold face.

5.1.1 *Single Relation Ranking.* Let Set_i and Set_j be the set of linguistic features of anchors A_i and A_j respectively. To evaluate Rel_k, we consider two characteristics: (1) the direction of relation Rel_k as encoded in the tree structure; and (2) the linguistic features, Set_i and Set_j, of anchors A_i and A_j. We need to construct multiple classifiers of single relation, one for each anchor pair of types C_i and C_j, to evaluate the relevance of Rel_k with respect to these two anchor types.

(a) *Construction of classifiers.* The training data for each classifier consists of anchor pairs of types C_i and C_j extracted from the training corpus. We use these anchor pairs to construct each classifier in four stages. First, we compose the set of possible patterns in the form $P^+ = \{P_m = <S_i\ –Rel–>\ S_j> \mid S_i \in Set_i, S_j \in Set_j\}$. The construction of P_m conforms to the two characteristics given above. Figure 4 illustrates several discourse and dependency patterns of P^+ constructed from a sample sentence.

Second, we identify the candidate anchor A, whose type matches slot C in a template.

Third, we find the correct patterns for the following two cases: (1) A_i, A_j are of correct anchor types; and (2) A_i is an action anchor, while A_j is a correct anchor. Any other patterns are considered incorrect. We note that the discourse and dependency paths between anchors A_i and A_j are either correct or wrong simultaneously.

Fourth, we evaluate the relevance of each pattern $P_m \in P^+$. Given the training set, let $PairSet_m$ be the set of anchor pairs extracted by P_m; and $PairSet^+(C_i, C_j)$ be the set of correct anchor pairs of types C_i, C_j. We evaluate both precision and recall of P_m as

$$Precision(P_m) = \frac{\|PairSet_m \cap PairsSet^+(C_i, C_j)\|}{\|PairSet_m\|} \quad (5)$$

$$Recall(P_m) = \frac{\|PairSet_m \cap PairsSet^+(C_i, C_j)\|}{\|PairsSet^+(C_i, C_j)\|} \quad (6)$$

These values are stored in the training model for use during testing.

(b) *Evaluation of a relation during testing.* Here we want to evaluate whether relation *InputRel* belongs to a path between anchors $InputA_i$ and $InputA_j$. We employ the constructed classifier for the anchor types $InputC_i$ and $InputC_j$ in two stages. First, we find a subset $P^{(0)} = \{P_m = <S_i$ *–InputRel->* $S_j> \in P^+ \mid\mid S_i \in InputSet_i, S_j \in InputSet_j\}$ of applicable patterns. Second, we utilize $P^{(0)}$ to find the pattern $P_m{}^{(0)}$ with maximal precision:

$$Precision\left(P_m^{(0)}\right) = argmax_{Pm \in P(0)} Precision(P_m) \quad (7)$$

A problem arises if $P_m{}^{(0)}$ is evaluated on only a small number of training instances. For example, we noticed that patterns that cover 1 or 2 instances may lead to $Precision = 1$, whereas on the testing corpus their accuracy becomes less than 50%. Therefore, it is important to also consider the recall parameter of $P_m{}^{(0)}$.

5.1.2 *Relation Path Ranking.* In this section, we want to evaluate the relation path connecting template slots C_i and C_j. We do this independently for each relation of type discourse and dependency. Let $Recall_k$ and $Precision_k$ be the recall and precision values of Rel_k in $Path = [A_i, Rel_1, \ldots, Rel_n, A_j]$, both obtained from the previous step. First, we calculate the average recall of the involved relations:

$$W = (1/Length_{Path}) * \Sigma_{Relk \in Path} Recall_k \quad (8)$$

W gives the average recall of the involved relations, and can be used as a measure of reliability of the relation $Path$. Next, we compute a combined score of average $Precision_k$ weighted by $Recall_k$:

$$Score = 1/(W * Length_{Path}) * \Sigma_{Relk \in Path} Recall_k * Precision_k \quad (9)$$

We use all $Precision_k$ values in the path here, since omitting a single relation may turn a correct path into the wrong one, or vice versa. The combined score value is used as a ranking of the relation path. Experiments show that we need to give priority to scores with higher reliability W. Hence we use the component of *(W, Score)* to evaluate each $Path$.

To illustrate relation path ranking, we consider the excerpt "named <AT-A1>**Marshal**</> played by Amourie Kats, who discovers all kinds of on and scary things going on in <AT-A2>**a seemingly quiet little town**</>". The discourse and dependency path evaluations are given in Tables III and IV,

Table III. The Evaluation of Discourse Path

			$\mathbf{W}_C$	$\mathbf{Score}_C$
list_personWord	<-elaboration-	pos_NN	0.05	0.29
Arg2	<-span-	ArgM-Loc	0.21	0.81
cand_ata1	-elaboration->	ArgM-Loc	0.22	0.86
cand_ata1	-elaboration->	ArgM-Loc	0.22	0.86
Overall			0.18	0.80

Table IV. Evaluation of a Dependency Path (which is long and thus unreliable)

			$\mathbf{W}_D$	$\mathbf{Score}_D$
Minipar_obj	<-obj-	town	0.05	0.33
Spade_Satellite	<-i-	pos_NN	0.16	0.09
Pos_NNP	<-fc-	pos_NN	0.11	0.03
list_personWord	<-pnmod-	pos_NN	0.06	0.19
list_personWord	<-pred-	pos_NN	0.1	0.23
Spade_Satellite	<-i-	pos_NN	0.16	0.09
cand_ata1	<-null-	town	0.04	0.49
cand_ata1	-null->	town	0.04	0.49
Minipar_obj	-rel->	list_frequent	0.1	0.13
cand_ata1	-i->	town	0.04	0.49
list_personWord	-mod->	town	0.04	0.49
list_personWord	-pcomp-n->	town	0.04	0.49
Overall			0.09	0.21

respectively. The discourse path obtained a relatively high score of $Score_C = 0.8$, whereas the dependency path score of $Score_D = 0.21$ is relatively low. Thus, in the following section, we need to decide which score is more reliable.

5.2 Combination of Different Relation Path Types

At this point, we evaluated the set of relations between anchors using the discourse and dependency classifiers. Our next task is to combine the binary evaluations of relations in order to form the templates. The architecture for combining the relations is given in Figure 5.

First, we extract anchors and exhaustively pair them. Second, we identify possible relation paths between anchor pairs and eliminate those relation paths that can be removed reliably by using the negative discourse and negative dependency classifiers. Third, we evaluate the remaining anchor pairs using the positive discourse and dependency classifiers. Finally, we obtain the integral evaluation of anchor pairs and exhaustively merge them into templates.

5.2.1 *Filtering Wrong Paths Between Anchors.* Due to the limited amount of training data, there are insufficient positive examples to learn good classifiers to identify possible paths between anchors. This is partly due to the large variations in linguistic structures, and thus "good" combinations of features tend to coexist in multiple negative instances as well. Moreover, if a relation path is long, the prediction based on such a path is not reliable. As an example, we consider the sentence *"The <Role-A1> gore</> <Role-A2> campaign</> has been embarrassed by accusations that so many absentee ballots possibly for*

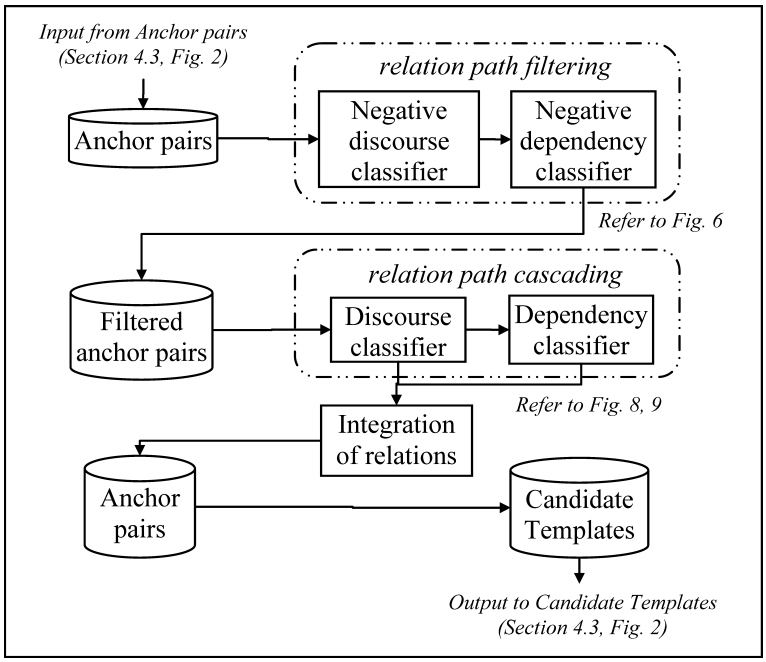

Fig. 5. Combination of binary relation evaluations into templates. Anchor pairs undergo relation path filtering, relation path cascading, and integration of relations.

mr. bush were set aside." Linguistic features of the anchors *"gore"* and *"bush"* are similar, so linguistic features are not sufficient to distinguish the correct paths. Additionally, the relation *"<-nn-"* not only exists in the correct dependency path *"gore <-nn- campaign"*, but also in numerous negative dependency paths such as *"bush <-pcomp-n- _ <-mod- _ <-mod-before- _ <-i- _ <-fc- _ -s-> _ -nn-> gore"*.

To overcome the above problem, we need to utilize the large number of negative path examples in the training process. The idea is in finding relation paths in negative examples that are sufficiently reliable to reject the relation path. Intuitively, the idea of rejecting paths is similar to the rule of disjunction negation in De Morgan's theorem. We may think of relations in relation paths as a set of boolean variables $X_1,\ldots,X_n.$, so that X_i becomes *"true"* if and only if $Relation_i$ may occur in some path between anchors A_i and A_j. In order to accept this path as correct, we need to check whether the combination X_1, ..., X_n can be accepted (this checking may be approximated using statistical models). However, it is possible to simplify our task if we try to prove that a path between anchors is wrong. In this case, we only need to reliably identify any relation in any path as incorrect according to the rule $\overline{X_1 \wedge \cdots \wedge X_n} = \overline{X_1} \vee \cdots \vee \overline{X_n}$(as illustrated in Figure 6). Therefore, if we can reliably reject a single relation in one of the paths, we may reject the integral relation path.

We implemented the idea of rejecting paths by constructing the set of *negative classifiers* for each of the relation types: discourse, semantic, and dependency. Our construction method is the same as that for positive

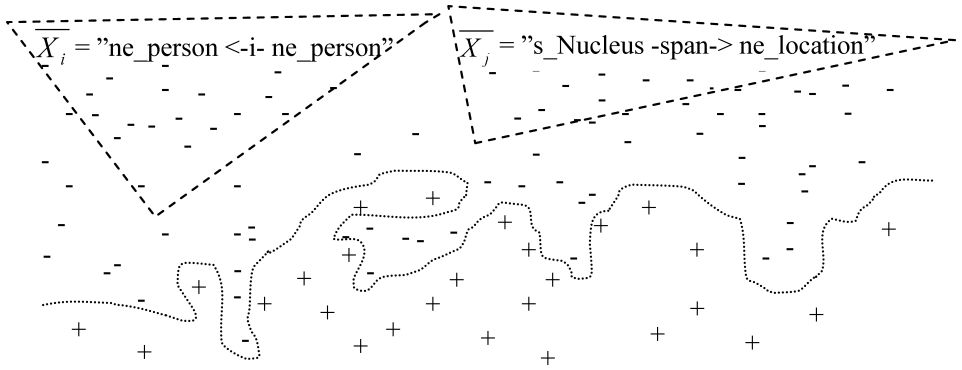

Fig. 6. Filtering wrong paths using negative patterns. While it is difficult to find the precise separation line between positive and negative paths, precise filtering conditions are coarse and easy to obtain.

classifiers (defined in Section 5.1) with two major differences, which address two problems. The first problem is that a negative classifier needs to extract negative relation paths. To do so, we converted positive paths into negative ones, and vice versa. Similar to the construction of positive classifiers, the set of anchor pairs of types C_i and C_j is represented in the form $P^+ = \{P_m = <S_i \text{ –}Rel\text{->} S_j> \mid S_i \in Set_i, S_j \in Set_j\}$. In this formula, Set_i and Set_j denote linguistic features of anchors A_i and A_j, respectively. To extract the negative relation paths, we estimate the pattern P_m as *correct for negative classifiers* if at least one of its anchors does not match its respective slot C_i in a template T.

The second problem is that a small loss in precision in the negative classifiers may lead to a large loss of positive paths, because the number of negative paths may be several times higher. We coped with this problem at two levels.

At the first level, we accept a pattern P_m only if $Precision(P_m) = 1$. This is done to minimize the number of false alarms for negative classifiers.

At the second level, we reject negative paths only if all the negative classifiers reject them. This allows us to minimize mistakes at several levels of discourse, that is, semantic and dependency relations. As an illustration, we consider the sentence *"What happened here today was of enormous importance to both political <RoleA2>**parties**</> because the <RoleA1>**democrats**</> had said all along that this was the place that there were enough recounted votes that they could gather to win the presidency*". The sentence contains the correct relation *Role* between the words *"democrats"* and *"parties"*. However, from the excerpt of the dependency parse tree given in Figure 7, we found that the dependency path is long and not reliable in making the prediction. If we consider the evaluations by negative classifiers for this instance, the negative dependency classifier rejects the path *"democrats <-s- <-i- <-comp1- <-mod- _ -pred-> _ -pcomp-n-> parties"* because this path contains the interclausal relation "*<-i-*" and the corresponding pattern *"Cand_rolea1 <-i- Cand_rolea2"* with a high value of $\boldsymbol{Score}_D = 1$. On the other hand, the negative discourse classifier does not reject the path *"democrats <-attribution- _ <-explanation- _ -span-> parties"*, and therefore this path is selected for further analysis by the positive classifiers.

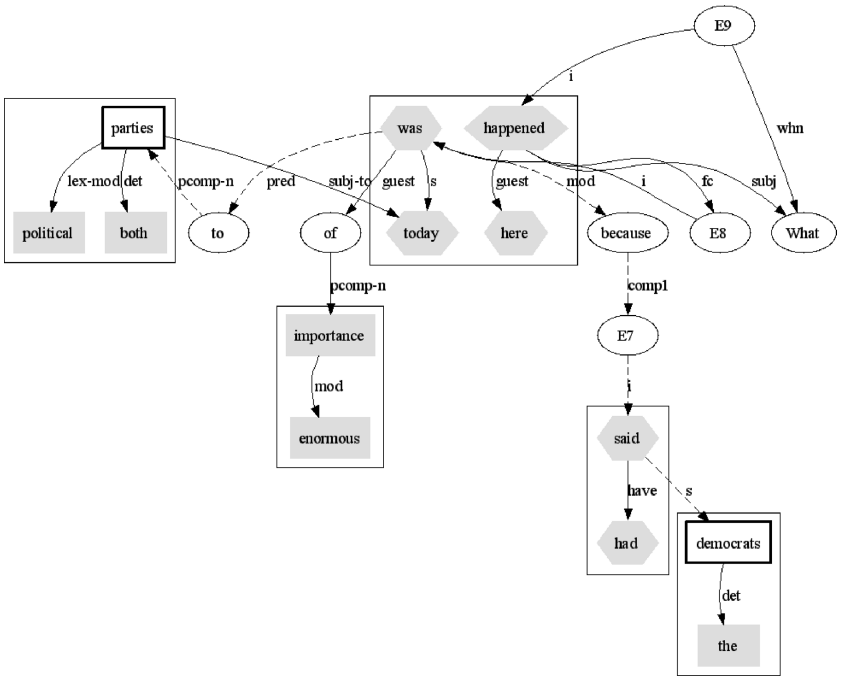

Fig. 7. Example of a dependency path in the role relation.

5.2.2 *Cascading of Classifiers.* At least one-third of the incoming dependency paths have ≥3 relations. We refer to such paths as *long paths*. In the case of long paths, the dependency analysis is insufficient to distinguish the correct paths from the wrong ones. Thus, long paths adversely affect the performance of the dependency classifier [Maslennikov et al. 2006]. Here, we conjecture that cascading the discourse and dependency classifiers should reduce data sparseness and improve the overall performance. The construction of the discourse and dependency classifiers was described earlier in Section 5.1.

As an example, we consider the positive instance "*It's a dark comedy about a boy named <AT-A1>Marshal</> played by Amorie Kats who discovers all kinds of on and scary things going on in <AT-A2>a seemingly quiet little town</>". The long dependency path "Marshal <–obj- _ <-i- _ <-fc- _ <-pnmod- _ <-pred- _ <-i- _ <-null- _ -null-> _ -rel-> _ -i-> _ -mod-> _ -pcomp-n-> town*" contains multiple noisy relations "*<-i-*" or "*<-null-*", which normally indicate that the relation path should be rejected. Hence, if the long dependency path is correct, it is important to add discourse features to relation patterns in the dependency classifier. For this purpose, we created the cascading algorithm shown in Figure 8.

The input to the cascading algorithm consists of the set of anchor pairs *PairSet*(C_i, C_j) and relation paths between these anchors. The output of the algorithm consists of the set of evaluated dependency paths between anchors A_i and A_j. The cascading process is based on the possible output of the discourse classifier. If the discourse classifier does not extract the path between anchors

Step 1. Extract discourse relation paths using the discourse classifier. Let $PairSet_C(C_i, C_j)$ be the set of extracted anchor pairs.
  a. For each anchor pair A_i and A_j in $PairSet_C(C_i, C_j)$, obtain the discourse relation path $Path_C = [A_i, Rel_1, ..., Rel_n, A_j]$ between the anchors.
  b. Compose a relation set $RelSet_C = \{Rel_1, ..., Rel_n\}$ from $Path_C$.

Step 2. Add the relation set $RelSet_C$ to the set of linguistic features of the anchors A_i and A_j: $Set_i' = Set_i \cup RelSet_C$, $Set_j' = Set_j \cup RelSet_C$; where Set_i and Set_j are the original sets of linguistic features of A_i and A_j.

Step 3. Extract dependency relation paths $PairSet_D'(C_i, C_j)$ with the dependency classifier, using the sets of extended linguistic features Set_i' and Set_j'.

Fig. 8. Algorithm for cascading classifiers.

A_i and A_j, no features are input into the dependency classifier. Otherwise, if the discourse classifier extracts a relation path $Path_C$ between A_i and A_j, the cascading algorithm extends the sets of linguistic features Set_i and Set_j for each pair of anchors A_i and A_j: $Set_i' = Set_i \cup RelSet_C$, $Set_j' = Set_j \cup RelSet_C$. The extended sets Set_i', Set_i' are used as input linguistic features to the dependency classifier. In contrast to Set_i and Set_j, Set_i' and Set_i' explicitly capture the information about useful discourse relations.

The cascading process is illustrated in Figure 9. For the previous example sentence, the discourse classifier extracted the relation *"elaboration"*. This relation was added to the dependency classifier as a linguistic feature *"c_elaboration"*. Consequently, some of the generated dependency patterns contain the relation "<-c_elaboration-". Since such patterns cover only the correct long dependency paths, the noisy relations of "*<-i-*" or "*<-null-*" are not included in the dependency patterns for short paths. Therefore, the sparseness of dependency relations on long paths does not affect the performance of the dependency classifier on short paths.

5.2.3 *Integrating Different Relation Path Types.* The purpose of this stage is to integrate the evaluations for different types of relation paths. The input to this stage consists of the evaluated relation paths $Path_C$ and $Path_D$ for discourse and dependency relations respectively. Let $(W_l, Score_l)$ be an evaluation for $Path_l$, $l \in [c, d]$. We first define an integral path $Path_I$ between A_i and A_j as (1) $Path_I$ is enabled if at least one of $Path_l$, $l \in [c, d]$, is enabled; and (2) $Path_I$ is correct if at least one of $Path_l$ is correct.

To evaluate $Path_I$, we first consider the average recall W_l of each $Path_l$, since W_l estimates the reliability of $Score_l$. We define a weighted average for $Path_l$ as

$$W_I = W_C + W_D \tag{10}$$

$$Score_I = 1/W_I * \Sigma_l W_l * Score_l \tag{11}$$

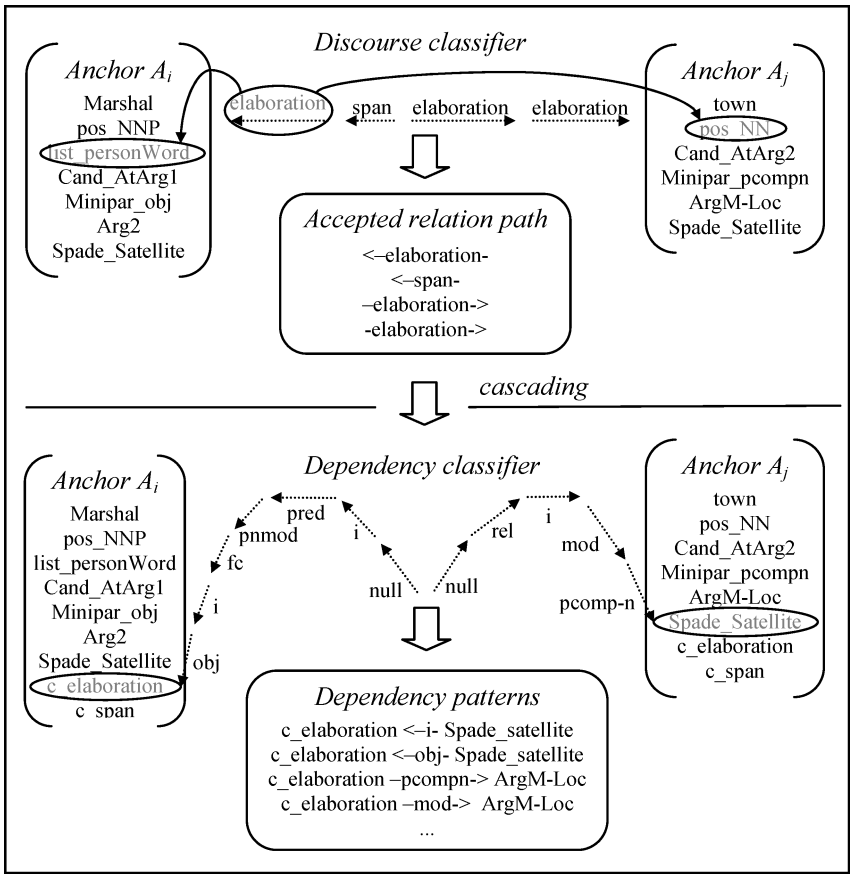

Fig. 9. Cascading the discourse and dependency classifiers. The relations from the accepted discourse paths are input as features into the dependency classifier.

Next, we estimate the threshold score $Score_I^O$ above which $Score_I$ is acceptable. This score may be found by analyzing the integral paths on the training corpus. Let $S_I = \{Path_I\}$ be the set of integral paths between anchors A_i and A_j on the training set. Among the paths in S_I, we need to define a set function $S_I(X) = \{Path_I \mid Score_I(Path_I) \geq X\}$ and find the optimal threshold for X. We find the optimal threshold based on the F_1-measure, since precision and recall are equally important in information extraction. Let $S_I(X)^+ \subset S_I(X)$ and $S(X)^+ \subset S(X)$ be the sets of correct path extractions. Let $F_I(X)$ be the F_1-measure of $S_I(X)$:

$$P_I(X) = \frac{\| S_I(X)^+ \|}{\| S_I(X) \|} \tag{12}$$

$$R_I(X) = \frac{\| S_I(X)^+ \|}{\| S(X)^+ \|} \tag{13}$$

$$F_I(X) = \frac{2^* P_I(X)^* R_I(X)}{P_I(X) + R_I(X)} \tag{14}$$

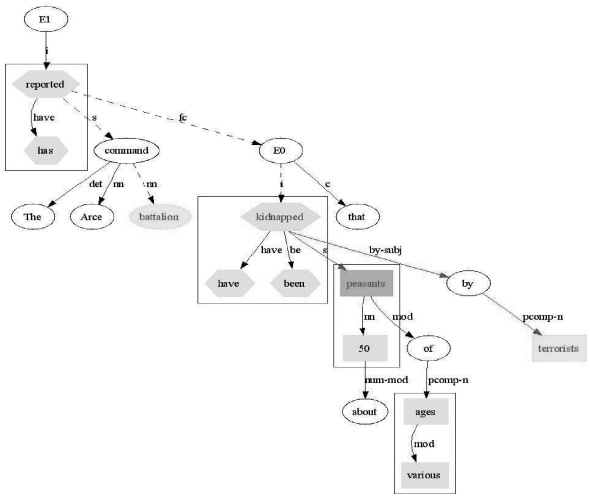

Fig. 10. Dependency parsing of a sample sentence. The positive relation paths between the anchors are highlighted in red.

Based on the computed values $F_I(X)$ for each X on the training data, we determine the optimal threshold as $Score_I^O = argmax_X F_I(X)$, which corresponds to the maximal expected F_1-measure of anchor pair A_i and A_j.

5.3 Evaluation of Templates

At this stage, we have a set of accepted integral relation paths between any anchor pair A_i and A_j. The next task is to merge an appropriate set of anchors into candidate templates and evaluate these templates. As an example, we extracted four anchors (in bold font) from the sentence *"The Arce cand_perp:**battalion** command has reported that about 50 cand_victim:**peasants** of various ages have been cand_action:**kidnapped** by cand_perp:**terrorists**"*. In order to construct the candidate templates, we considered three rationales: (1) the anchor type must match the slot type in the candidate template; (2) all anchors in the candidate template must be connected by the integral paths; and (3) if there is an empty slot in a template and it is possible to fill it with an anchor under condition (2), then this slot must be filled. These rationales significantly reduce the search space to the magnitude of 10–100, and hence, it is possible to check the space of candidate templates exhaustively.

The construction of candidate templates is illustrated in Figure 10 and Table V. For simplicity, we assume that the integral paths have the same topology as the dependency paths in Figure 10. There are three anchor pairs that satisfy condition (2): "battalion -> kidnapped," "kidnapped -> terrorists," and "kidnapped -> peasants". Based on rationales (1) and (3), we obtain the two candidate templates shown in Table V.

Here is the formal explanation on how the candidate templates in Table V are derived. For each sentence, we compose a set of candidate templates T using the extracted relation paths between each A_i and A_j. Let $\{C = C_1,..,C_K\}$ be the set of slot types in T, and $\{A = A_1,..,A_L\}$ be the set of extracted

Table V. Candidate Templates

Perpetrator	*Action*	*Victim*	*Target*
battalion	kidnapped	peasants	-
terrorists	kidnapped	peasants	-

anchors. First, we regroup anchors A according to their respective types. Let $A^{(k)} = \{A_1^{(k)}, \ldots, A_{L_k}^{(k)}\}$ be the projection of A onto the type C_k, $\forall k \in N$, $k \leq K$. Let $T = A^{(1)} \times A^{(2)} \times .. \times A^{(K)}$ be the set of possible template fillings (as in Table V). The elements of T are denoted $T_1, \ldots, T_M$, where every $T_i \in T$ is represented as $T_i = \{A_i{}^{(1)}, \ldots, A_i{}^{(K)}\}$. Our aim is to evaluate T and find the optimal filling $T_0 \in T$. For this purpose, we use the previously calculated scores of relation paths between every two anchors A_i and A_j.

Our constructed method needs to tackle two problems: (1) choose the optimal template filling $T_0 \in T$ within each sentence by performing local optimization of the chosen anchors at the sentence level; and (2) reject T_0 if it is not reliable. For this purpose, we need to perform the global optimization based on all the available training data.

Additionally, we do not consider a template filling T_i, in which some anchor A_i has no accepted integral relation path with each other anchor A_j in the same template filling. Due to this optimization, it is possible to check the set of template fillings T exhaustively for each sentence (in most cases, the size of T does not exceed 100).

5.3.1 *Local Optimization of Template Fillings.* Based on the previously defined $Score_I(A_i, A_j)$ in Eq. (11), it is possible to rank all the fillings in T. The following are the rationales for our approach to ranking: (a) we should give equal priority to the cases when all the anchor pair scores $Score_I(A_i, A_j)$ are equal for all $<i, j>$, versus when $Score_I(A_i, A_j)$ differ significantly; and (b) we prefer a model that combines scores $Score_I(A_i, A_j)$ explicitly. The condition (a) is violated if we multiply the scores $Score_I(A_i, A_j)$, while, in our opinion, condition (b) is not fully satisfied for the case of more complicated models of relational learning such as that described in Culotta and Sorensen [2004]. Thus, we decided to construct the formula using the weighted sum of $Score_I(A_i, A_j)$. First, for each filling $T_i \in T$ we calculate the aggregate relation score for all the involved anchor pairs:

$$Relation\_Score_S(T_i) = \frac{\sum_{1 \leq i,j \leq K} Score_I(A_i, A_j)}{M} \tag{15}$$

where K is the number of extracted slots and M is the number of extracted relation paths between anchors A_i and A_j.

Next, we combine each $Entity\_Score(A_k)$ from Eq. (1) to obtain the overall entity score of template T_i. We calculate the extracted entity score based on the scores of all the anchors in T_i:

$$Entity\_Score_S(T_i) = \sum\nolimits_{1 \leq k \leq K} Entity\_Score(A_k)/K \tag{16}$$

Each $Entity\_Score(A_k)$ is obtained from Eq. (1).

Finally, we obtain the combined evaluation for a template:

$$Score_S(T_i) = \lambda * Entity\_Score_S(T_i) + (1 - \lambda) * Relation\_Score_S(T_i) \quad (17)$$

where λ is a predefined constant.

Considering the example template fillings in Table V, the integral relation path *"terrorists –> kidnapped"* is ranked higher than the path *"battalion –> kidnapped"*. Hence, at this stage we ranked the template filling *<terrorists, kidnapped, peasants>* higher than the template filling *<battalion, kidnapped, peasants>*.

5.3.2 *Global Optimization of Template Fillings.* In order to decide whether the template T_i should be accepted or rejected, we need to determine a threshold $Score_T^{\circ}$ from the training data. If the anchors of a candidate template match the slots in a correct template, we consider the candidate template to be correct. Let $TrainT = \{T_i\}$ be the set of candidate templates extracted from the training data, $TrainT^+ \subset TrainT$ be the subset of correct candidate templates, and $TotalT^+$ be the total set of correct templates in the training data. Also, let $TrainT(X) = \{T_i \mid Score_T(T_i) \geq X, T_i \in TrainT\}$ be the set of candidate templates with a score above X, and $TrainT^+(X) \subset TrainT(X)$ be the subset of correct candidate templates. We define the measures of precision, recall, and F_1 as follows:

$$P_T(X) = \frac{\parallel TrainT^+(X) \parallel}{\parallel TrainT(X) \parallel} \quad (18)$$

$$R_T(X) = \frac{\parallel TrainT^+(X) \parallel}{\parallel TotalT^+ \parallel} \quad (19)$$

$$F_T(X) = \frac{2 * P_T(X) R_T(X)}{P_T(X) + R_T(X)} \quad (20)$$

Since performance in IE is measured in F_1-measure, an appropriate threshold for the most suitable candidate templates is

$$Score_T^O = argmax_X F_T(X) \quad (21)$$

The value $Score_T^O$ is used as a training model. During testing, we accept a candidate template $InputT_i$ if $Score_T(InputT_i) > Score_T^O$.

As an example, we consider the template fillings in Table V. The top ranked template filling <terrorists, kidnapped, peasants> is accepted as correct at this stage of selection.

6. EXPERIMENTAL RESULTS

In order to verify that our approach is scalable, we designed experiments on several tasks such as the extraction of template slots (Message Understanding Conference 4 (MUC4), terrorism domain), template events (Message Understanding Conference 6 (MUC6), management succession domain), and relations (Automatic Content Extraction 2003 competition, Relation Detection and Characterization (ACE RDC 2003) task).

The first domain, MUC4, is composed of a set of news articles about terrorism events in the Latin America region. The training set of this domain consists

Table VI. General and Specific Types in ACE RDC 2003

Type	Subtypes
Role	Management, General-Staff, Member, Owner, Founder, Affiliate-Partner, Client, Citizen-Of, Other
Part	Subsidiary, Part-Of, Other
At	Located, Based-In, Residence
Near	Relative-Location
Soc	Parent, Sibling, Spouse, Grandparent

of 1,300 articles, while the testing set has 200 articles labeled TST3 and TST4. Approximately half of the articles in MUC4 contain an event to be extracted, while the other half has no relevant information. We measured the performance of ARE for the slot-based task, as defined by Riloff [1996]. The target slots for this task consist of Perpetrator, Victim, and Target.

The second domain, MUC6, is composed of Wall Street Journal articles about business news. Since all relation parsers that we used (Minipar, ASSERT, and Spade) can work only at the level of single sentence, we decided to process the MUC6 domain at the sentence level as well. Thus, we followed the methodology of Soderland [1999], where the performance was measured only at the level of single sentence. The modified MUC6 corpus by Soderland [1999] consists of 599 training documents and 100 testing documents. Some of the news refers to management succession events, which we had to extract. To extract this information automatically into a database, it is important to identify the respective company, position of the manager, and the name of a retired and/or hired person. Thus, a typical event consists of slots for PersonIn, PersonOut, Company, and Post. Additionally, this task consists of two parts: (1) extracting event templates; and (2) merging these templates across sentences. Slots that match some templates are counted as correct, otherwise they are counted as wrong. In case of an extra slot extracted in a template, all the slots in such a template are counted as incorrect extractions.

The third domain is composed of texts from the training corpora of the Automatic Content Extraction evaluation. Our task is to extract explicit relations, following the Relation Detection and Characterization task in this evaluation for the year 2003. For brevity, we refer to this task as ACE RDC 2003. The relations in this task consist of 5 major types and 24 specific subtypes. The list of general types and their subtypes is given in Table VI. Since the evaluation data for ACE RDC 2003 is not available under the copyright policy, previous studies (e.g., Zhang et al. [2006]) used only the training part of the ACE RDC 2003 corpora and subdivided it into a training and testing set. In our experiments we followed this methodology, and used only the English portion of the ACE RDC 2003 training data. We retained 97 documents as testing data and the remaining 155 documents for training.

6.1 Evaluation Metrics

For all the domains, we utilize a precision P, recall R, and F_1 as measures of performance. These measures may be defined via computing the correlation between the extracted, correct, and wrong instances. Let N_{ext} denote the number

of extracted instances, N_{corr} be the number of correctly extracted instances, and N_{all} be the number of correct instances in a dataset. P, R, and F_1 are defined as follows:

$$P = N_{corr}/N_{ext} \tag{22}$$

$$R = N_{corr}/N_{all} \tag{23}$$

$$F_1 = 2 * P * R/(P + R) \tag{24}$$

Metrics in (22) to (24) are used as the basis for evaluating the quality of extracted templates in all the domains MUC4, MUC6, and ACE RDC. However, there are significant differences in evaluation methodology for each of the domains. The performance on the MUC4 domain is measured for each slot individually; the MUC6 task measures performance on the extracted templates; while the ACE RDC 2003 task evaluates performance on matching relations. To reconcile these differences, we constructed templates for all the domains and measured the required type of performance for each domain. Our artificial templates for the ACE RDC 2003 task consist of only two slots, which correspond to entities of correct relations.

6.2 Experimental Setups

There are three main objectives of our experimental evaluation. First, we want to examine the impact of several hypotheses on the overall performance of ARE, on whether: (1) the addition of discourse relations helps handle the long-distance dependency paths between the anchors; (2) the filtering of negative paths is important in extracting the correct paths between the anchors; and (3) the cascading of discourse and dependency classifiers separates the long and short dependency paths well. Second, we want to compare the performance of ARE with the results of other state-of-art supervised systems on 100% of the training data. Third, we analyze the errors incurred and suggest possible directions to overcome these errors.

Our evaluation of the ARE system is based on several settings: (1) *ARE(Dep)*, which uses only dependency relations; (2) *ARE(Int)* with the integration of discourse and semantic relations in addition to dependency relations; (3) *ARE(Int+Neg)*: which is *ARE(Int)* with the addition of the filtering strategy; and (4) *ARE(Full)*: which combines *ARE(Int+Neg)* together with the cascading strategy. For each of the domains, we will compare the performance of the ARE system with the relevant state-of-the-art approaches. We also report the p-values for the 2-paired statistical significance test.

6.3 Results For MUC4

We compare our results on the MUC4 domain with GRID by Xiao et al. [2004]. The performance of ARE and GRID is compared on 100% of the training data. The results are given in Table VII.

We make several observations about the performance of ARE:

(A) The comparative results for different ARE settings are shown in the shaded cells of Table VII. Notably, the overall improvement from

Table VII. Results for MUC4

System	P	R	F_1
GRID	62%	52%	57%
ARE(Dep)	58%	63%	60%
ARE(Int)	65%	61%	63%
ARE(Int+Neg)	66.6%	61.9%	64.2%
ARE(Full)	66.8%	63.1%	64.9%

ARE(Dep) to *ARE(Full)* reaches 4.9%. It is achieved consecutively by the addition of discourse relations, filtering, and cascading strategies in *ARE(Int)*, *ARE(Int+Neg)*, and *ARE(Full)*, respectively. The improvement of *ARE(Int)* over *ARE(Dep)*, *ARE(Int+Neg)* over *ARE(Int)*, and *ARE(Full)* over *ARE(Int+Neg)* are statistically significant for the p-values of $P = 2.1\%$, $P = 3.6\%$, and $P = 4.1\%$ in the F_1-measure, respectively. There are several reasons for such improvement. First, discourse relations help handle the long-distance dependency relations. For example, splitting the instance *"[X distributed leaflets] [claiming responsibility for murder of Y]"* leads to correct extraction of X as Perpetrator. This is because (a) the intraclausal analysis extracts reliable dependency relation *"murder -> of -> Y"*; and (b) the interclausal analysis is important for the additional extraction of reliable clausal relation path *"X <-span- _ -Elaboration-> murder"*. Second, cascading the discourse and dependency classifiers helps to combine important features into a single pattern. For example, the extracted relation from the discourse classier for the path *"X <-span- _ -Elaboration-> murder"* consist of *"<-span-"* and *"-Elaboration->"*. These relations help to generate meaningful dependency patterns such as "discourse_span -mod-> pos_NN". Thus, highly precise discourse patterns help to increase the recall of dependency patterns with average precision. Therefore, the discourse patterns can be considered as meta-rules in the terminology of Ciravegna [2001].

(B) The F_1 performance of *ARE(Full)* improved significantly over the best state-of-the-art system *GRID* is shown in bold face. The use of dependency relations in *ARE(Dep)* achieves an improvement of 3% over the *GRID* system. This improvement demonstrates that both the anchor extraction and dependency relation extraction stages are useful. While anchor extraction enables the selection of prominent candidates, the use of the dependency relation makes the learning model flexible to possible alignment and paraphrasing of instances. The overall improvement in F_1 reaches 7.9%.

(C) Several errors were incurred in this domain. We noticed that (1) a few Perpetrator or Victim anchors were missed because of mistakes in the Named Entity component; and (2) some Action anchors were missed as they are relatively sparse (e.g., the anchor 'destroyed'). Additionally, several mistakes were made in the semantic parsing component because of support verb constructions. For example, the instance *"X conducted murder of Y"* was parsed under the frame 'conducted' instead of the relevant

Table VIII. Results for MUC6

System	P	R	F_1
Chieu et al. '02	75%	49%	59%
ARE(Dep)	73%	58%	65%
ARE(Int)	73%	70%	72%
ARE(Int+Neg)	75.2%	71.0%	73.0%
ARE(Full)	75.2%	73.1%	74.1%

frame 'murder'. Another serious reason for mistakes is a lack of connection between sentences. As an example, we consider the following sentences. *"The <Perpetrator>2 individuals</> broke into the offices and entered <Victim>gen. Leigh</>'s private office. The individuals used 9 mm semi-automatic pistols. According to the experts, <Victim>gen. Ruiz</> was hit 3 times and <Victim>gen Gustavo Leigh</> was hit five times."* In these sentences, the experts reported an event and the event really happened. However, the extraction of the event in these sentences depends on whether we can trust the people who reported this event. A small change of the word 'experts' into 'these individuals' leads to no slots in these sentences. Therefore, it is important to perform the discourse analysis of the whole text to extract the slots reliably.

6.4 Results on MUC6

Our performance on the MUC6 domain is given in Table VIII. We make several observations based on our experiments:

(A) The results of different ARE settings are presented in the shaded cells of Table VIII. We noticed that the addition of discourse relations in *ARE(Int)* improved the F_1 performance of *ARE(Dep)* by 7%, and it is statistically significant ($p = 1.4\%$). Similar to MUC4, the major reason for this improvement is the better handling of long-distance dependency relations between anchors. Next, the filtering strategy in *ARE(Int+Neg)* gives an additional improvement of 1% in the F_1 performance (with $p = 4.3\%$). For example, multiple negative paths such as *"Org:Fox <-s- named -mod-> E0 -i-> Action:succeed"* were eliminated due to the use of negative classifiers. Finally, the cascading of discourse and dependency classifiers in *ARE(Full)* improved the F_1 performance by a further 1% (with $p = 3.3\%$). The major reason for this improvement is that discourse analysis is able to remove wrong dependency paths from the training data. Moreover, as the lexicon for MUC6 is relatively stable, the respective improvement of *ARE(Full)* is larger than that for MUC4.

Additionally, linguistic phenomena which are not covered by semantic, discourse, or dependency analysis can be overcome on other layers. As an example, we noticed that some mistakes in semantic parsing were caused by light verb constructions (LVC). For example, the sentence *"The sell-off followed the resignation late Monday of Jamie Kellner, the president of Fox Broadcasting Co., yesterday"* contains LVC *"followed the resignation"*, from which the ASSERT parser extracted the target "followed" instead of the

Table IX. Results for ACE RDC 2003, General Types

System	P	R	F_1
Zhang et al.'06	79%	66%	72%
ARE(Int)	79%	66%	73%
ARE(Int+Neg)	83.3%	66.9%	74.2%
ARE(Full)	81.4%	69.3%	74.8%

target "resignation". However, due to the correct extraction of the anchor "resignation", the dependency classifier was able to extract the correct succession event.

(B) The comparison between *ARE(Full)* and another state-of-the-art system, that is, Chieu et al. [2002] is highlighted in bold face, in which *ARE(Full)* achieves a significant 15% improvement in F_1 over Chieu et al. [2002]. The main reason for this vast improvement is that MUC6 contains only 28% of *hard case* instances (in which relation paths have length > 1 in terms of nouns and verbs), as reported by Maslennikov et al. [2006]. Therefore the majority of the dependency paths are short and predictive in this domain. In our opinion, it is the major reason for the 15.1% improvement over Chieu et al. [2002] using the *ARE(Full)* setting.

(C) We noticed that 22% of the correct test sentences contain two answer templates, and that entities in many such templates are intertwined. This is the major cause of mistakes on MUC6. As an illustration, we consider the sentence *"Stephan W. Cole, now president of American consumer products, will be chief operating officer."* ARE was able to extract "Stephan W. Cole" as both PersonOut and PersonIn anchor from this sentence. ARE was also able to correctly extract the template *"PersonIn:<Stephan W. Cole> Post:<chief operating officer>"*. However, ARE missed the template *"PersonOut:<Stephan W. Cole> Post:president Organization:<American consumer products>"* because it was unable to find the relevant action anchor indicating 'stepping down'. For this case, it is important to find some ways (e.g., domain knowledge) to interpret cues like "now" in this discourse.

6.5 Results for ACE RDC with General Types

The main characteristic of the ACE corpus is that it contains a large number of variations, most of which are incorrect. For instance, only 2% of possible dependency paths are labeled as correct. We noticed that 38% of relation paths in ACE contain a single relation, 28% contain 2 relations and 34% contain $\geq$3 relations. For the case of $\geq$3 relations, the analysis of dependency paths alone is not sufficient to eliminate the unreliable paths. Our results for general types and subtypes are presented in Table IX.

Several observations can be drawn from Table IX:

(A) The comparative F_1 performance of several ARE settings is given in the shaded cells. We noticed that the improvement of the filtering setting in *ARE(Int+Neg)* in comparison to the baseline setup *ARE(Int)* is 1.2%, with

Table X. Results for ACE RDC 2003, Specific Types

System	P	R	F_1
Zhang et al. (2006)	64%	51%	57%
ARE(Int)	67%	54%	61%
ARE(Int+Neg)	70.9%	54.7%	61.8%
ARE(Full)	72.1%	55.5%	62.7%

a significance value of p = 3.9%. The cascading strategy *ARE(Full)* further improved F_1 performance by 0.6% (with p = 4.7%). Therefore, the results show that our overall strategy enables several classifiers to support each other.

(B) As shown in bold face, the overall F_1 improvement of *ARE(Full)* against Zhang et al. [2006] reaches 2.8%. Notably, this improvement is smaller than that for MUC4 and MUC6. In our opinion, this is because the linguistic features that we used for ACE RDC are not sensitive enough to filter the anchors that are irrelevant to the extracted slot. For example, if we found the anchor 'succeeded' in the MUC6 domain, it is a likely evidence of the succession event. Similarly, anchors like 'terrorists' or 'kidnapped' are good clues for a possible terrorism event in the MUC-4 domain. However, in the ACE RDC 2003 task, even if we know that the anchor 'George W. Bush' is a named entity of type Person and 'Texas' is Location, they can still belong to the two possible types, Role and At.

(C) Thus it is much harder to overcome the errors in the ACE RDC domain than that in MUC4 or MUC6. One of the causes of errors is that relations in this domain require world knowledge to make the correct extraction. For example, the excerpt *"<RoleA1>He</> was inducted into the <RoleA2>Kentucky Journalism Hall of Fame</>"* requires interpretation that the predicate 'inducted' implies being a member (the fine-grained relation is 'RoleMember'). Second, some of the classes contain instances with very similar meanings. For example, neither our linguistic features nor relations are sufficient to reliably distinguish the following instances: *"<RoleA1>professor</> at <RoleA2>Bauman State University</>"* and *"<AtA1>analysts</> at <AtA2>the FBI crime lab</>"*.

6.6 Results for ACE RDC with Specific Types

The major characteristic of the ACE RDC task on specific relations is the high dimensionality, since there are 48 possible anchor types. For an average sentence that contains 5 entities, the number of possible entity pairs is 10 and the number of anchor pairs becomes 480, and each pair requires an analysis of the relation path. This indicates that discourse relations are crucial to reduce the possible variations to the level of a single clause.

From the results in Table X, we can draw the following observations:

(A) As shown in the shaded cells, the F_1 performance of the discourse-based strategy *ARE(Int)* is improved consecutively by the filtering *ARE(Ing+Neg)*

(p = 4.5%) and cascading *ARE(Full)* (p = 4.2%) strategies, which is statistically significant. The overall improvement from both strategies is 1.7% over *ARE(Int)*. It is comparable to the respective increases for the MUC4, MUC6, and ACE RDC 2003 (general types) domains. Therefore, the filtering and cascading strategies are useful in coping with the data sparseness problem.

(B) The setting *ARE(Full)* improved over the state-of-the-art system by Zhang et al. [2006] by 5.7% in terms of the F_1 measure. This improvement may be attributed to two reasons: (1) the discourse analysis is useful because the anchors may belong to different clauses; and (2) cascading the discourse and dependency classifiers helps in cases of long-distance dependency relation paths.

(C) The errors in this domain occur for the following reasons. First, 155 training texts may not be sufficient to distinguish the 24 fine-grained relations in this domain, as in the instances *"<NearRelativeLocationA1>Landstuhl</> is set amid rolling <NearRelativeLocationA2>hills</>"* and *"<AtResidenceA1>She</>'d come <AtResidenceA2>home</> drunk"*. In our opinion, in this case it is important to use world knowledge extensively. Second, some of the smallest classes were missed in our randomly chosen training texts, such as the class "SocialRelativeLocation". Finally, many mistakes occurred because of the similarity between several relation classes. For example, it is unclear to us which features can be used to differentiate the instance *"<RoleGeneralStaffA2>Russian</> <RoleGeneralStaffA1>officials</>"* from the instance *"<RoleCitizenOfA2> Iranian</> <RoleCitizenOfA1>officials </>"*.

6.7 Remarks on Performance

Here we comment on the scalability of our method over large datasets.

(A) The efficiency of ARE components can be summarized as follows: (a) the dependency parser such as Minipar by Lin [1998] is rather efficient; it can parse 300 words per second on the Pentium II 300 with 128MB memory.[2] (b) There are efficient solutions for semantic parsing. For example, Collobert and Weston [2007] reported the performance of 0.02 second per sentence; this is 250 times faster than the ASSERT parser, while the performance of both semantic parsers is comparable. (c) The NE recognition systems such as that of Niu et al. [2003] are generally considered efficient. (d) The implementation of ARE requires $O(N_{instances} * log\ N_{instances})$ actions to induce and sort patterns in relation path classifiers, where $N_{instances}$ denotes the number of training instances.

(B) The current implementation of ARE is relatively slow. The parsing of an average sentence takes about 14 seconds during the training stage, and 6 seconds during the testing stage on a Pentium IV HT with 1 Gb memory.

[2]http://www. cs.ualberta. ca/~lindek/minipar.htm.

In our opinion, the major reason for the inefficiency in processing is that we use a relatively slow language, Perl.

7. CONCLUSION

The sparseness of instances in free text is one of the crucial problems in performing information extraction. In this article, we hypothesized that it is important to tackle this problem at several levels of language discourse. Therefore, we integrated discourse, semantic, and dependency relations in a single multiresolution framework called ARE (Anchors and Relations). The first part of this framework extracts key phrases (anchors) that are good candidates to fill template slots. The second part uses discourse and dependency/semantic classifiers to evaluate relation paths between the anchors. This framework outperformed the previous state-of-the-art systems for several news domains of MUC4 (terrorism), MUC6 (management succession), and ACE RDC 2003 (general news). Notably, the improvement on MUC6 over the previous state-of-the-art system is 15% in F_1-measure.

There are two problems facing ARE. First, the number of negative relation paths between the anchors is several times bigger than that for positive relation paths. To tackle this problem, we added the strategy that filters negative relation paths. It improved the performance in all the domains, with a maximum improvement of 1.2% in MUC4 and ACE RDC 2003 on specific types. Second, the performance of the dependency relation path classifier on long dependency paths is significantly worse than that for short dependency paths. To tackle this problem, we cascaded the discourse classifier with the dependency classifier to separate long dependency paths from short dependency paths, which improved the performance by about 1.1% on MUC6.

The experimental results revealed that ARE is portable to different free text datasets and domains without human effort, provided that external components can be used to extract linguistic features and to perform parsing. Moreover, the ARE approach combines dependency, semantic, and discourse analysis so that mistakes in texts can be handled at several levels. In our opinion, this reduces the need for human annotation of semi-structured texts or texts with informal grammar and terminology such as blogs or Web extracted data [Bunescu and Mooney 2007].

Overall, our results suggest that effective handling of the data sparseness problem at different levels of discourse will greatly enhance performance. For future work, we propose several approaches to improve performance. First, we plan to explore in-depth methods to integrate world knowledge in ontologies. Second, it is important to improve the performance of individual parsers in order to avoid mistakes in parsing such as the incorrect handling of supporting verb constructions. Third, it may be important to add parsers that will handle other NLP phenomena. For example, adding syntactic parsing may lead to further improvement in the overall performance of ARE. Fourth, we plan to exploit large datasets in the semi-supervised framework in our future work.

APPENDIX

Table XI. Parameters of ARE that Require Tuning

Parameter	Value	Tuning	Section
δ_{EM}	0.1	manual	4.1, (3)
ω	0.4	manual	4.1 (4)
λ	0.4	manual	5.3.1, (17)

REFERENCES

AGICHTEIN, E. AND GRAVANO, L. 2000. Snowball: Extracting relations from large plain text collections. In *Proceedings of the ACM Conference on Digital Libraries*. ACM, New York, 85–94.

BRIN, S. 1998. Extracting patterns and relations from the World Wide Web. In *Proceedings of the WebDB Workshop at EDBT*.

BUNESCU, R. AND MOONEY, R. 2007. Learning to extract relations from the web using minimal supervision. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL'07)*. 576–583.

CIMIANO, P., REYLE, U., AND SARIC, J. 2005. Ontology-driven discourse analysis for information extraction. *Data Knowl. Engin. 55*, 1, 59–83.

CIMIANO, P. AND REYLE, U. 2003. Ontology-based semantic construction, under specification and disambiguation. In *Proceedings of the Prospects and Advances in the Syntax-Semantic Interface Workshop*.

CHIEU, H. L. AND NG, H. T. 2002. A maximum entropy approach to information extraction from semi-structured and free text, In *Proceedings of the 18th National Conference on Artificial Intelligence (AAAI'02)*. 786–791.

CIRAVEGNA, F. 2001. Adaptive information extraction from text by rule induction and generalization. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI'01)*.

COLLOBERT, R. AND WESTON, J. 2007. Fast semantic extraction using a novel neural network architecture. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (ACL'07)*. 232–239.

CUI, H., KAN, M.Y., AND CHUA, T. S. 2004. Unsupervised learning of soft patterns for definitional question answering. In *Proceedings of the 13th International Conference on World Wide Web (WWW'04)*. 90–99.

CUI, H., KAN, M.Y., AND CHUA, T. S. 2005. Question answering passage retrieval using dependency relations. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development of Information Retrieval (SIGIR'05)*. ACM, New York, 400–407.

CULOTTA, A. AND SORENSEN, J. 2004. Dependency tree kernels for relation extraction. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*.

DAVIDOV, D., RAPPOPORT, A., AND KOPPEL, M. 2007. Fully unsupervised discovery of concept-specific relationships by web mining. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (ACL'07)*. 232–239.

DEMPSTER, A., LAIRD, N., AND RUBIN, D. 1977. Maximum likelihood from incomplete data via the EM algorithm. *J. Royal Stat. Soc. B 39*, 1, 1–38.

EFRON, B. 1979. Bootstrap methods: Another look at the Jackknife. *Ann. Stat.* 7, 1, 1–26.

ETZIONI, O., CAFARELLA, M., DOWNEY, D., POPESCU, A.M., SHAKED, T., SODERLAND, S., WELD, D., AND YATES, A. 2005. Unsupervised named-entity extraction from the Web: An experimental study. *Artif. Intell. 165*, 1, 91–134.

GROSZ, B. AND SIDNER, C. 1986. Attention, intentions and the structure of discourse. *Comput. Linguist. 12*, 3, 175–204.

HALLIDAY, M. AND HASAN, R. 1976. *Cohesion in English*. Longman, London.

LIN, D. 1997. Dependency-based evaluation of Minipar. In *Proceedings of the Workshop on the Evaluation of Parsing Systems*.

LIN, W., YANGARBER, R., AND GRISHMAN, R. 2003. Bootstrapped learning of semantic classes from positive and negative examples. In *Proceedings of the ICML Workshop on the Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining*. 103–110.

LIU, B., LEE, W. S., YU, P. S., AND LI, X. 2002. Partially supervised classification of text documents. In *Proceedings of the Nineteenth International Conference on Machine Learning (ICML)*. 387–394.

MASLENNIKOV, M. AND CHUA, T. S. 2007. A Multi-resolution framework for information extraction from free text. In *Proceedings of the 45th Annual Meeting on Association for Computational Linguistics (ACL'07)*. 592–599.

MASLENNIKOV, M., GOH, H. K., AND CHUA, T. S. 2006. ARE: Instance splitting strategies for dependency relation-based information extraction. In *Proceedings of the 44th Annual Meeting on Association for Computational Linguistics (ACL'06)*. 571–578.

MILTSAKAKI, E. 2003. The syntax-discourse interface: effects of the main-subordinate distinction on attention structure. Ph.D. dissertation, University of Pennsylvania.

MOENS, M. F. AND DE BUSSER, R. 2002. First steps in building a model for the retrieval of court decisions. *Int. J. Human-Comput. Stud. 57*, 5, 429–446.

NIU, C., LI, W., DING J., AND SRIHARI, R. K. 2003. A bootstrapping approach to named entity classification using successive learners. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics (ACL'03)*. 335–342.

PRADHAN, S., WARD, W., HACIOGLU, K., MARTIN, J., AND JURAFSKY, D. 2004. Shallow semantic parsing using support vector machines. In *Proceedings of the Human Language Technology Conference/North American Chapter of the Association of Computational Linguistics (HLT/NAACL'04)*.

RILOFF, E., WIEBE, J., AND PHILLIPS, W. 2005. Exploiting subjectivity classification to improve information extraction. In *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI'05)*. 1106–1111.

RILOFF, E. AND JONES, R. 1999. Learning dictionaries for information extraction by multi-level bootstrapping. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence and the Eleventh Innovative Applications of Artificial Intelligence Conference (AAAI'99/IAAI'99)*. 474–479.

RILOFF, E. 1996. Automatically generating extraction patterns from untagged text. In *Proceedings of the 13th National Conference on Artificial Intelligence (AAAI'96)*,1044–1049.

ROTH, D. AND YIH, W. 2002. Probabilistic reasoning for entity and relation recognition. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING'02)*, 1–7.

SODERLAND, S. 1999. Learning information extraction rules for semi-structured and free text. *Mach. Learn. 34*, 1–3, 233–272.

SORICUT, R. AND MARCU, D. 2003. Sentence level discourse parsing using syntactic and lexical information. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (NAACL'03)*. 149–156.

SURDEANU, M., HARABAGIU, S., WILLIAMS, J., AND AARSETH, P. 2003. Using predicate arguments structures for information extraction, In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics (ACL'03)*. 8–15.

TABOADA, M. AND MANN, W. 2005. Applications of rhetorical structure theory. *Discourse Stud. 8*, 4, 567–588.

THELEN, M. AND RILOFF, E. 2002. A bootstrapping method for learning semantic lexicons using extraction pattern context. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'02)*. 214–221.

WEBBER, B., STONE, M., JOSHI, A., AND KNOTT, A. 2002. Anaphora and discourse structure. *Comput. Linguist. 29*, 4.

XIAO, J., CHUA, T. S., AND LIU, J. 2003. A global rule induction approach to information extraction. In *Proceedings of the 15th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'03)*. 530–536.

XIAO, J., CHUA, T. S., AND CUI, H. 2004. Cascading use of soft and hard matching pattern rules for weakly supervised information extraction. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING'04)*.

YANG, H., CUI, H., KAN, M.Y., MASLENNIKOV, M., QIU, L. AND CHUA, T. S. 2003. QUALIFIER in TREC-12 QA main task. In *Notebook of the 12th Text Retrieval Conference (TREC'03)*.

YANGARBER, R., LIN, W., AND GRISHMAN, R. 2002. Unsupervised learning of generalized names. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING'02)*. 1–7.

YANGARBER, R. 2003. Counter-training in discovery of semantic patterns. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics (ACL'03)*. 343–350.

ZHANG, M., ZHANG, J., SU, J., AND ZHOU, G. 2006. A composite kernel to extract relations between entities with both flat and structured features. In *Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics (ACL'06)*. 825–832.

ZHOU, G., SU, J., AND ZHANG, M. 2006. Modeling commonality among related classes in relation extraction. In *Proceedings of 44th Annual Meeting of the Association for Computational Linguistics (ACL'06)*. 121–128.

Received November 2007; revised May 2009; accepted July 1009