ARE: Instance Splitting Strategies for Dependency Relation-based Information Extraction

Mstislav Maslennikov **Hai-Kiat Goh** **Tat-Seng Chua**

Department of Computer Science
School of Computing
National University of Singapore
{maslenni, gohhaiki, chuats}@ comp.nus.edu.sg

Abstract

Information Extraction (IE) is a fundamental technology for NLP. Previous methods for IE were relying on co-occurrence relations, soft patterns and properties of the target (for example, syntactic role), which result in problems of handling paraphrasing and alignment of instances. Our system ARE (Anchor and Relation) is based on the dependency relation model and tackles these problems by unifying entities according to their dependency relations, which we found to provide more invariant relations between entities in many cases. In order to exploit the complexity and characteristics of relation paths, we further classify the relation paths into the categories of 'easy', 'average' and 'hard', and utilize different extraction strategies based on the characteristics of those categories. Our extraction method leads to improvement in performance by 3% and 6% for MUC4 and MUC6 respectively as compared to the state-of-art IE systems.

1 Introduction

Information Extraction (IE) is one of the fundamental problems of natural language processing. Progress in IE is important to enhance results in such tasks as Question Answering, Information Retrieval and Text Summarization. Multiple efforts in MUC series allowed IE systems to achieve near-human performance in such domains as biological (Humphreys et al., 2000), terrorism (Kaufmann, 1992; Kaufmann, 1993) and management succession (Kaufmann, 1995).

The IE task is formulated for MUC series as filling of several predefined slots in a template. The terrorism template consists of slots Perpetrator, Victim and Target; the slots in the management succession template are Org, PersonIn, PersonOut and Post. We decided to choose both terrorism and management succession domains, from MUC4 and MUC6 respectively, in order to demonstrate that our idea is applicable to multiple domains.

Paraphrasing of instances is one of the crucial problems in IE. This problem leads to data sparseness in situations when information is expressed in different ways. As an example, consider the excerpts "Terrorists attacked victims" and "Victims *were* attacked *by unidentified* terrorists". These instances have very similar semantic meaning. However, context-based approaches such as Autoslog-TS by Riloff (1996) and Yangarber et al. (2002) may face difficulties in handling these instances effectively because the context of entity 'victims' is located on the left context in the first instance and on the right context in the second. For these cases, we found that we are able to verify the context by performing dependency relation parsing (Lin, 1997), which outputs the word 'victims' as an object in both instances, with 'attacked' as a verb and 'terrorists' as a subject. After grouping of same syntactic roles in the above examples, we are able to unify these instances.

Another problem in IE systems is word alignment. Insertion or deletion of tokens prevents instances from being generalized effectively during learning. Therefore, the instances "Victims *were* attacked *by* terrorists" and "Victims *were recently* attacked *by* terrorists" are difficult to unify. The common approach adopted in GRID by Xiao et al. (2003) is to apply more stable chunks such as noun phrases and verb phrases. Another recent approach by Cui et al. (2005) utilizes soft patterns for probabilistic matching of tokens. However, a longer insertion leads to a more complicated structure, as in the instance "Victims, living near the shop, *went out for a walk and were* attacked *by* terrorists". Since there may be many inserted words, both approaches may also be inefficient for this case. Similar to the paraphrasing problem, the word alignment problem may be handled with dependency relations in many cases. We found that the relation subject-verb-object for words 'victims', 'attacked' and 'terrorists' remains invariant for the above two instances.

Before IE can be performed, we need to identify sentences containing possible slots. This is

Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions, pages 571–578,
Sydney, July 2006.

done through the identification of cue phrases which we call *anchors* or *anchor cues*. However, natural texts tend to have diverse terminologies, which require semantic features for generalization. These features include semantic classes, Named Entities (NE) and support from ontology (for example, synsets in Wordnet). If such features are predefined, then changes in terminology (for instance, addition of new terrorism organization) will lead to a loss in recall. To avoid this, we exploit automatic mining techniques for anchor cues. Examples of anchors are the words "terrorists" or "guerrilla" that signify a possible candidate for the Perpetrator slot.

From the reviewed works, we observe that the inefficient use of relations causes problems of paraphrasing and alignment and the related data sparseness problem in current IE systems. As a result, training and testing instances in the systems often lack generality. This paper aims to tackle these problems with the help of dependency relation-based model for IE. Although dependency relations provide invariant structures for many instances as illustrated above, they tend to be efficient only for short sentences and make errors on long distance relations. To tackle this problem, we classify relations into 'simple', 'average' and 'hard' categories, depending on the complexity of the dependency relation paths. We then employ different strategies to perform IE in each category.

The main contributions of our work are as follows. First, we propose a dependency relation based model for IE. Second, we perform classification of instances into several categories based on the complexity of dependency relation structures, and employ the action promotion strategy to tackle the problem of long distance relations.

The remaining parts of the paper are organized as follows. Section 2 discusses related work and Section 3 introduces our approach for constructing ARE. Section 4 introduces our method for splitting instances into categories. Section 5 describes our experimental setups and results and, finally, Section 6 concludes the paper.

2 Related work

There are several research directions in Information Extraction. We highlight a few directions in IE such as case frame based modeling in PALKA by Kim and Moldovan (1995) and CRYSTAL by Soderland et al. (1995); rule-based learning in Autoslog-TS by Riloff et al. (1996); and classification-based learning by Chieu et al. (2002). Although systems representing these directions have very different learning models, paraphrasing and alignment problems still have no reliable solution.

Case frame based IE systems incorporate domain-dependent knowledge in the processing and learning of semantic constraints. However, concept hierarchy used in case frames is typically encoded manually and requires additional human labor for porting across domains. Moreover, the systems tend to rely on heuristics in order to match case frames. PALKA by Kim and Moldovan (1995) performs keyword-based matching of concepts, while CRYSTAL by Soderland et al. (1995) relied on additional domain-specific annotation and associated lexicon for matching.

Rule-based IE models allow differentiation of rules according to their performance. Autoslog-TS by Riloff (1996) learns the context rules for extraction and ranks them according to their performance on the training corpus. Although this approach is suitable for automatic training, Xiao et al. (2004) stated that hard matching techniques tend to have low recall due to data sparseness problem. To overcome this problem, $(LP)^2$ by Ciravegna (2002) utilizes rules with high precision in order to improve the precision of rules with average recall. However, $(LP)^2$ is developed for semi-structured textual domain, where we can find consistent lexical patterns at surface text level. This is not the same for free-text, in which different order of words or an extra clause in a sentence may cause paraphrasing and alignment problems respectively, such as the example excerpts "terrorists attacked peasants" and "peasants *were* attacked *2 months ago by* terrorists".

The classification-based approaches such as by Chieu and Ng (2002) tend to outperform rule-based approaches. However, Ciravegna (2001) argued that it is difficult to examine the result obtained by classifiers. Thus, interpretability of the learned knowledge is a serious bottleneck of the classification approach. Additionally, Zhou and Su (2002) trained classifiers for Named Entity extraction and reported that performance degrades rapidly if the training corpus size is below 100KB. It implies that human experts have to spend long hours to annotate a sufficiently large amount of training corpus.

Several recent researches focused on the extraction of relationships using classifiers. Roth and Yih (2002) learned the entities and relations together. The joint learning improves the performance of NE recognition in cases such as "X killed Y". It also prevents the propagation of mistakes in NE extraction to the extraction of relations. However, long distance relations between entities are likely to cause mistakes in relation extraction. A possible approach for modeling relations of different complexity is the use of dependency-based kernel trees in support vector machines by Culotta and Sorensen (2004). The authors reported that non-relation instances are very heterogeneous, and

hence they suggested the additional step of extracting candidate relations before classification.

3 Our approach

Differing from previous systems, the language model in ARE is based on dependency relations obtained from Minipar by Lin (1997). In the first stage, ARE tries to identify possible candidates for filling slots in a sentence. For example, words such as 'terrorist' or 'guerrilla' can fill the slot for Perpetrator in the terrorism domain. We refer to these candidates as *anchors* or *anchor cues*. In the second stage, ARE defines the dependency relations that connect anchor cues. We exploit dependency relations to provide more invariant structures for similar sentences with different syntactic structures. After extracting the possible relations between anchor cues, we form several possible parsing paths and rank them. Based on the ranking, we choose the optimal filling of slots.

Ranking strategy may be unnecessary in cases when entities are represented in the SVO form. Ranking strategy may also fail in situations of long distance relations. To handle such problems, we categorize the sentences into 3 categories of: simple, average and hard, depending on the complexity of the dependency relations. We then apply different strategies to tackle sentences in each category effectively. The following subsections discuss details of our approach.

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	-

Table 1. Linguistic features for anchor extraction

Every token in ARE may be represented at a different level of representations, including: Lexical, Part-of-Speech, Named Entities, Synonyms and Concept classes. The synonym set and concept classes are mainly obtained from Wordnet. We use NLProcessor from Infogistics Ltd for the extraction of part-of-speech, noun phrases and verb phrases (we refer to them as *phrases*). Named Entities are extracted with the program used in Yang et al. (2003). Additionally, we employed the co-reference module for the extraction of meaningful pronouns. It is used for linking entities across clauses or sentences, for example in "John *works in* XYZ Corp. He *was* appointed *as a* vice-president *a month ago*" and could achieve an accuracy of 62%. After preprocessing and feature extraction, we obtain the linguistic features in Table 1.

3.1 Mining of anchor cues

In order to extract possible anchors and relations from every sentence, we need to select features to support the generalization of words. This generalization may be different for different classes of words. For example, person names may be generalized as a Named Entity PERSON, whereas for 'murder' and 'assassinate', the optimal generalization would be the concept class 'kill' in the WordNet hypernym tree. To support several generalizations, we need to store multiple representations of every word or token.

Mining of anchor cues or anchors is crucial in order to unify meaningful entities in a sentence, for example words 'terrorists', 'individuals' and 'soldiers' from Table 1. In the terrorism domain, we consider 4 types of anchor cues: Perpetrator, Action, Victim, and Target of destruction. For management succession domain, we have 6 types: Post, Person In, Person Out, Action and Organization. Each set of anchor cues may be seen as a pre-defined semantic type where the tokens are mined automatically. The anchor cues are further classified into two categories: general type *A* and action type *D*. Action type anchor cues are those with verbs or verb phrases describing a particular action or movement. General type encompasses any predefined type that does not fall under the action type cues.

In the first stage, we need to extract anchor cues for every type. Let P be an input phrase, and A_j be the anchor of type j that we want to match. The similarity score of P for A_j in sentence S is given by:

$$\begin{aligned} Phrase\_Score_S(P,A_j) = {} & \delta_1 * S\_lexical_S(P,A_j + \delta_2 * S\_POS_S(P,A_j) \\ & + \delta_3 * S\_NE_S(P,A_j) + \delta_4 * S\_Syn_S(P,A_j) \\ & + \delta_5 * S\_Concept\text{-}Class_S(P,A_j) \end{aligned} \quad (1)$$

where $S\_XXX_S(P,A_j)$ is a score function for the type A_j and δ_i is the importance weight for A_j. In order to extract the score function, we use entities from slots in the training instances. Each $S\_XXX_S(P,A_j)$ is calculated as a ratio of occurrence in positive slots versus all the slots:

$$S\_XXX_S(P,A_j) = \frac{\#(P \text{ in positive slots of the type } A_j)}{\#(\text{all slots of the type } A_j)} \quad (2)$$

We classify the phrase P as belonging to an anchor cue A of type j if $Phrase\_Score_S(P,A_j) \geq \omega$, where ω is an empirically determined threshold. The weights $\vec{\delta} = (\delta_1, ..., \delta_5)$ are learned automatically using Expectation Maximization by Dempster et al. (1977). Using anchors from training instances as ground truth, we iteratively input different sets of weights into EM to maximize the overall score.

Consider the excerpts "Terrorists attacked victims", "Peasants were murdered by unidentified individuals" and "Soldiers participated in massacre of Jesuit priests". Let W_i denotes the position of token i in the instances. After mining of anchors, we are able to extract meaningful anchor cues in these sentences as shown in Table 2:

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			

Table 2. Instances with anchor cues

3.2 Relationship extraction and ranking

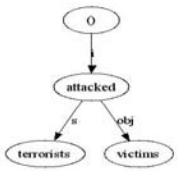


Figure 1. Dependency tree

In the next stage, we need to find meaningful relations to unify instances using the anchor cues. This unification is done using dependency trees of sentences. The dependency relations for the first sentence are given in Figure 1.

From the dependency tree, we need to identify the SVO relations between anchor cues. In cases when there are multiple relations linking many potential subjects, verbs or objects, we need to select the best relations under the circumstances. Our scheme for relation ranking is as follows.

First, we rank each single relation individually based on the probability that it appears in the respective context template slot in the training data. We use the following formula to capture the quality of a relation Rel which gives higher weight to more frequently occurring relations:

$$Quality\ (Rel\ , A_1, A_2) = \frac{\sum_{\bar{S}} \| \{R_i \mid R_i \in R, R_i = Rel\ \} \|}{\sum_{\bar{S}} \| \{R_i \mid R_i \in S_i\} \|} \quad (3)$$

where $\bar{S}$ is a set of sentences containing relation *Rel*, anchors A_1 and A_2; R denotes relation path connecting A_1 and A_2 in a sentence S_i; $||X||$ denotes size of the set X.

Second, we need to take into account the entity height in the dependency tree. We calculate height as a distance to the root node. Our intuition is that the nodes on the higher level of dependency tree are more important, because they may be linked to more nodes or entities. The following example in Figure 2 illustrates it.

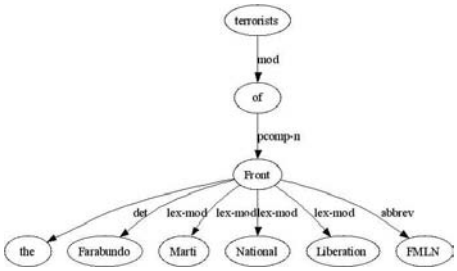


Figure 2. Example of entity in a dependency tree

Here, the node 'terrorists' is the most representative in the whole tree, and thus relations nearer to 'terrorists' should have higher weight. Therefore, we give a slightly higher weight to the links that are closer to the root node as follows:

$$Height_s(Rel) = log_2(Const - Distance(Root, Rel)) \quad (4)$$

where *Const* is set to be larger than the depth of nodes in the tree.

Third, we need to calculate the score of relation path $R_{i->j}$ between each pair of anchors A_i and A_j, where A_i and A_j belong to different anchor cue types. The path score of $R_{i->j}$ depends on both quality and height of participating relations:

$$Score_s(A_i, A_j) = \Sigma_{Ri \in R} \{Height_s(R_i) * Quality(R_i)\} / Length_{ij} \quad (5)$$

where $Length_{ij}$ is the length of path $R_{i->j}$. Division on $Length_{ij}$ allows normalizing *Score* against the length of $R_{i->j}$. The formula (5) tends to give higher scores to shorter paths. Therefore, the path ending with 'terrorist' will be preferred in the previous example to the equivalent path ending with 'MRTA'.

Finally, we need to find optimal filling of a template T. 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 anchors. First, we regroup anchors A according to their respective types. Let $A^{(k)} = \{A_1^{(k)}, ..., A_{L_k}^{(k)}\}$ be the projection of A onto the type C_k, $\forall k \in N$, $k \le K$. Let $F = A^{(1)} \times A^{(2)} \times .. \times A^{(K)}$ be the set of possible template fillings. The elements of F are denoted as $F_1, .., F_M$, where every $F_i \in F$ is represented as $F_i = \{A_i^{(1)}, .., A_i^{(K)}\}$. Our aim is to evaluate F and find the optimal filling $F_0 \in F$. For this purpose, we use the previously calculated scores of relation paths between every two anchors A_i and A_j.

Based on the previously defined $Score_S(A_i, A_j)$, it is possible to rank all the fillings in F. For each filling $F_i \in F$ we calculate the aggregate score for all the involved anchor pairs:

$$Relation\_Score_s(F_i) = \frac{\sum_{1 \le i, j \le K} Score_s(A_i, A_j)}{M} \quad (7)$$

where K is number of slot types and M denotes the number of relation paths between anchors in F_i.

After calculating $Relation\_Score_S(F_i)$, it is used for ranking all possible template fillings. The next step is to join entity and relation scores. We defined the entity score of F_i as an average of the scores of participating anchors:

$$Entity\_Score_s(F_i) = \sum_{1 \le k \le K} Phrase\_Score_s(A_i^{(k)}) / K \quad (8)$$

We combine entity and relation scores of F_i into the overall formula for ranking.

$$Rank_S(F_i) = \lambda * Entity\_Score_S(F_i) + (1-\lambda) * Relation\_Score_S(F_i) \quad (9)$$

The application of Subject-Verb-Object (SVO) relations facilitates the grouping of subjects,

verbs and objects together. For the 3 instances in Table 2 containing the anchor cues, the unified SVO relations are given in Table 3.

W_{-2}	W_{-1}	W_0	Instance is
Perp_Cue	attacked	Victim_Cue	+
Perp_Cue	murdered	Victim_Cue	+
Perp_Cue	participated	?	-

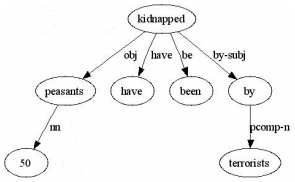
Table 3. Unification based on SVO relations

The first 2 instances are unified correctly. The only exception is the slot in the third case, which is missing because the target is not an object of 'participated'.

4 Category Splitting

Through our experiments, we found that the combination of relations and anchors are essential for improving IE performance. However, relations alone are not applicable across all situations because of long distance relations and possible dependency relation parsing errors, especially for long sentences. Since the relations in long sentences are often complicated, parsing errors are very difficult to avoid. Furthermore, application of dependency relations on long sentences may lead to incorrect extractions and decrease the performance.

Through the analysis of instances, we noticed that dependency trees have different complexity for different sentences. Therefore, we decided to classify sentences into 3 categories based on the complexity of dependency relations between the action cues (V) and the likely subject (S) and object cues (O). Category 1 is when the potential SVO's are connected directly to each other (simple category); Category 2 is when S or O is one link away from V in terms of nouns or verbs (average category); and Category 3 is when the path distances between potential S, V, and Os are more than 2 links away (hard category).



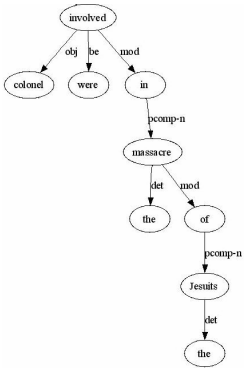


Figure 3. Simple category Figure 4. Average category

Figure 3 and Figure 4 illustrate the dependency parse trees for the simple and average categories respectively derived from the sentences: "50 peasants of have been kidnapped by terrorists" and "a colonel was involved in the massacre of the Jesuits". These trees represent 2 common structures in the MUC4 domain. By taking advantage of this commonality, we can further improve the performance of extraction. We notice that in the simple category, the perpetrator cue ('terrorists') is always a subject, action cue ('kidnapped') a verb, and victim cue ('peasants') an object. For the average category, perpetrator and victim commonly appear under 3 relations: subject, object and pcomp-n. The most difficult category is the hard category, since in this category relations can be distant. We thus primarily rely on anchors for extraction and have to give less importance to dependency parsing.

In order to process the different categories, we utilize the specific strategies for each category. As an example, the instance "X murdered Y" requires only the analysis of the context verb 'murdered' in the simple category. It is different from the instances "X *investigated* murder *of* Y" and "X *conducted* murder *of* Y" in the average category, in which transition of word 'investigated' into 'conducted' makes X a perpetrator. We refer to the anchor 'murder' in the first and second instances as *promotable* and *non-promotable* respectively. Additionally, we denote that the token 'conducted' is the optimal node for promotion of 'murder', whereas the anchor 'investigate' is not. This example illustrates the importance of support verb analysis specifically for the average category.

Algorithm

1) Analyze category
   If(simple)
   - Perform token reordering based on SVO relations
   Else if (average) *ProcessAverage*
   Else *ProcessHard*
2) Fill template slots

Function ProcessAverage

1) Find the nearest missing anchor in the previous sentences
2) Find the optimal linking node for action anchor in every F_i
3) Find the filling $F_i^{(0)} = \text{argmax}_i \text{Rank}(F_i)$
4) Use F_i for filling the template if $\text{Rank}_0 > \theta_2$, where θ_2 is an empirical threshold

Function ProcessHard

1) Perform token reordering based on anchors
2) Use linguistic+ syntactic + semantic feature of the head noun. Eg. Caps, 'subj', etc
3) Find the optimal linking node for action anchor in every F_i
4) Find the filling $F_i^{(0)} = \text{argmax}_i \text{Rank}(F_i)$
5) Use F_i for filling the template if $\text{Rank}_0 > \theta_3$, where θ_3 is an empirical threshold

Figure 5. Category processing

The main steps of our algorithm for performing IE in different categories are given in Figure 5. Although some steps are common for every category, the processing strategies are different.

Simple category

For simple category, we reorder tokens according to their slot types. Based on this reordering, we fill the template.

Average category

For average category, our strategy consists of 4 steps. First, in the case of missing anchor type we try to find it in the nearest previous sentence. Consider an example from MUC-6: "*Look at what happened to* John Sculley, *Apple Computer's* former chairman. *Earlier this month* he *abruptly* resigned *as* chairman *of troubled* Spectrum Information Technologies." In this example, a noisy cue 'he' needs to be substituted with "John Sculley", which is a strong anchor cue. Second, we need to find an optimal promotion of a support verb. For example, in "X *conducted* murder of Y", the verb 'murder' should be linked with X and in the excerpt "X *investigated* murder of Y", it should not be promoted. Thus, we need to make 2 steps for promotion: (a) calculate importance of every word connecting the action cue such as 'murder' and 'distributed' and (b) find the optimal promotion for the word 'murder'. Third, using the predefined threshold λ we cutoff the instances with irrelevant support verbs (e.g., 'investigated'). Fourth, we reorder the tokens in order to group them according to the anchor types.

The following algorithm in Figure 6 estimates the importance of a token *W* for type *D* in the support verb structure. The input of the algorithm consists of sentences $S_1 \ldots S_N$ and two sets of tokens V_{neg}, V_{pos} co-occurring with anchor cue of type D. V_{neg} and V_{pos} are automatically tagged as irrelevant and relevant respectively based on preliminary marked keys in the training instances. The algorithm output represents the importance value between 0 to 1.

CalculateImportance (W, D)
1) Select sentences that contain anchor cue D
2) Extract linguistic features of V_{pos}, V_{neg} and D
3) Train using SVM on instances (V_{pos},D) and instances (V_{neg},D)
4) Return Importance(W) using SVM

Figure 6. Evaluation of word importance

We use the linguistic features for *W* and *D* as given in Table 1 to form the instances.

Hard category

In the hard category, we have to deal with long-distance relations: at least 2 anchors are more than 2 links away in the dependency tree. Consequently, dependency tree alone is not reliable for connecting nodes. To find an optimal connection, we primarily rely on comparison between several possible fillings of slots based on previously extracted anchor cues. Depending on the results of such comparison, we chose the filling that has the highest score. As an example, consider the hard category in the excerpt "MRTA *today distributed leaflets claiming responsibility for the* murder *of former defense* minister Enrique Lopez Albujar". The dependency tree for this instance is given in Figure 7.

Although words 'MRTA', 'murder' and 'minister' might be correctly extracted as anchors, the challenging problem is to decide whether 'MRTA' is a perpetrator. Anchors 'MRTA' and 'minister' are connected via the verb 'distributed'. However, the word 'murder' belongs to another branch of this verb.

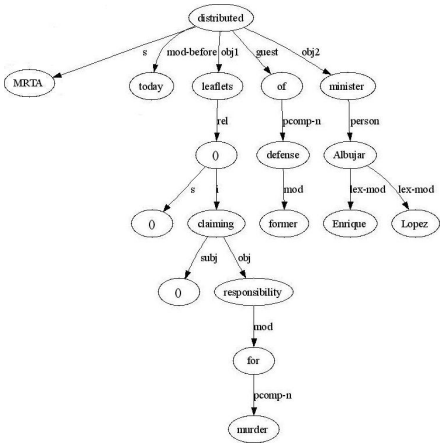


Figure 7. Hard case

Processing of such categories is challenging. Since relations are not reliable, we first need to rely on the anchor extraction stage. Nevertheless, the promotion strategy for the anchor cue 'murder' is still possible, although the corresponding branch in the dependency tree is long. Henceforth, we try to replace the verb 'distributed' by promoting the anchor 'murder'. To do so, we need to evaluate whether the nodes in between may be eliminated. For example, such elimination is possible in the pairs 'conducted' -> 'murder' and not possible in the pair 'investigated' -> 'murder', since in the excerpt "X *investigated* murder" X is not a perpetrator. If the elimination is possible, we apply the promotion algorithm given on Figure 8:

FindOptimalPromotion (F_i)
1) $Z = \varnothing$
2) For each $A_i^{(j1)}, A_i^{(j2)} \in F_i$
   $Z = Z \cup P_{j1->j2}$
   End_for
3) Output Top(Z)

Figure 8. Token promotion algorithm

The algorithm checks path $P_{j1->j2}$ that connect anchors $A_i^{(j1)}$ and $A_i^{(j2)}$ in the filling F_i; the nodes from $P_{j1->j2}$ are added to the set *Z*. Finally, the top node of the set *Z* is chosen as an optimal node for the promotion. The example optimal node for promotion of the word 'murder' on Figure 7 is the node 'distributed'.

Another important difference between the hard and average cases is in the calculation of $Rank_S(F_i)$ in Equation (9). We set $\lambda_{hard} > \lambda_{average}$ because long distance relations are less reliable in the hard case than in the average case.

5 Evaluation

In order to evaluate the efficiency of our method, we conduct our experiments in 2 domains: MUC4 (Kaufmann, 1992) and MUC6 (Kaufmann, 1995). The official corpus of MUC4 is released with MUC3; it covers terrorism in the Latin America region and consists of 1,700 texts. Among them, 1,300 documents belong to the training corpus. Testing was done on 25 relevant and 25 irrelevant texts from TST3, plus 25 relevant and 25 irrelevant texts from TST4, as is done in Xiao et al. (2004). MUC6 covers news articles in Management Succession domain. Its training corpus consists of 1201 instances, whereas the testing corpus consists of 76 person-ins, 82 person-outs, 123 positions, and 79 organizations. These slots we extracted in order to fill templates on a sentence-by-sentence basis, as is done by Chieu et al. (2002) and Soderland (1999).

Our experiments were designed to test the effectiveness of both case splitting and action verb promotion. The performance of ARE is compared to both the state-of-art systems and our baseline approach. We use 2 state-of-art systems for MUC4 and 1 system for MUC6. Our baseline system, *Anc+rel*, utilizes only anchors and relations without category splitting as described in Section 3. For our ARE system with case splitting, we present the results on *Overall* corpus, as well as separate results on *Simple*, *Average* and *Hard* categories. The *Overall* performance of ARE represents the result for all the categories combined together. Additionally, we test the impact of the action promotion (in the right column) for the average and hard categories.

Case (%)	Without promotion			With promotion		
	P	**R**	**F_1**	**P**	**R**	**F_1**
GRID	58%	56%	57%	-	-	-
Riloff'05	46%	52%	48%	-	-	-
Anc+rel (100%)	58%	59%	58%	58%	59%	58%
Overall (100%)	57%	60%	59%	58%	61%	60%
Simple (13%)	79%	86%	82%	79%	86%	82%
Average (22%)	64%	70%	67%	67%	71%	69%
Hard (65%)	50%	52%	51%	51%	53%	52%

Table 4. Results on MUC4 with case splitting

The comparative results are presented in Table 4 and Table 5 for MUC4 and MUC6, respectively. First, we review our experimental results on MUC4 corpus without promotion (left column) before proceeding to the right column.

a) From the results on Table 4 we observe that our baseline approach *Anc+rel* outperforms all the state-of-art systems. It demonstrates that both anchors and relations are useful. Anchors allow us to group entities according to their semantic meanings and thus to select of the most prominent candidates. Relations allow us to capture more invariant representation of instances. However, a sentence may contain very few high-quality relations. It implies that the relations ranking step is fuzzy in nature. In addition, we noticed that some anchor cues may be missing, whereas the other anchor types may be represented by several anchor cues. All these factors lead only to moderate improvement in performance, especially in comparison with GRID system.

b) *Overall*, the splitting of instances into categories turned out to be useful. Due to the application of specific strategies the performance increased by 1% over the baseline. However, the large dominance of the hard cases (65%) made this improvement modest.

c) We notice that the amount of variations for connecting anchor cues in the *Simple* category is relatively small. Therefore, the overall performance for this case reaches F_1=82%. The main errors here come from missing anchors resulting partly from mistakes in such component as NE detection.

d) The performance in the *Average* category is F_1=67%. It is lower than that for the simple category because of higher variability in relations and negative influence of support verbs. For example, for excerpt such as "X *investigated* murder *of* Y", the processing tends to make mistake without the analysis of semantic value of support verb 'investigated'.

e) *Hard* category achieves the lowest performance of F_1=51% among all the categories. Since for this category we have to rely mostly on anchors, the problem arises if these anchors provide the wrong clues. It happens if some of them are missing or are wrongly extracted. The other cause of mistakes is when ARE finds several anchor cues which belong to the same type.

Additional usage of promotion strategies allowed us to improve the performance further.

f) Overall, the addition of promotion strategy enables the system to further boost the performance to F_1=60%. It means that the promotion strategy is useful, especially for the average case. The improvement in comparison to the state-of-art system GRID is about 3%.

g) It achieved an F_1=69%, which is an improvement of 2%, for the *Average* category. It implies that the analysis of support verbs helps in revealing the differences between the instances such as "X *was involved in* kidnapping *of* Y" and "X *reported* kidnapping *of* Y".

h) The results in the *Hard* category improved moderately to F_1=52%. The reason for the improvement is that more anchor cues are captured after the promotion. Still, there are 2 types of common mis-

takes: 1) multiple or missing anchor cues of the same type and 2) anchors can be spread across several sentences or several clauses in the same sentence.

Case (%)	Without promotion			With promotion		
	P	R	F_1	P	R	F_1
Chieu et al.'02	74%	49%	59%	-	-	-
Anc+rel (100%)	78%	52%	62%	78%	52%	62%
Overall (100%)	72%	58%	64%	73%	58%	65%
Simple (45%)	85%	67%	75%	87%	68%	76%
Average (27%)	61%	55%	58%	64%	56%	60%
Hard (28%)	59%	44%	50%	59%	44%	50%

Table 5. Results on MUC6 with case splitting

For the MUC6 results given in Table 5, we observe that the overall improvement in performance of ARE system over Chieu et al.'02 is 6%. The trends of results for MUC6 are similar to that in MUC4. However, there are few important differences. First, 45% of instances in MUC6 fall into the S*imple* category, therefore this category dominates. The reason for this is that the terminologies used in Management Succession domain are more stable in comparison to the Terrorism domain. Second, there are more anchor types for this case and therefore the promotion strategy is applicable also to the simple case. Third, there is no improvement in performance for the *Hard* category. We believe the primary reason for it is that more stable language patterns are used in MUC6. Therefore, dependency relations are also more stable in MUC6 and the promotion strategy is not very important. Similar to MUC4, there are problems of missing anchors and mistakes in dependency parsing.

6 Conclusion

The current state-of-art IE methods tend to use co-occurrence relations for extraction of entities. Although context may provide a meaningful clue, the use of co-occurrence relations alone has serious limitations because of alignment and paraphrasing problems. In our work, we proposed to utilize dependency relations to tackle these problems. Based on the extracted anchor cues and relations between them, we split instances into 'simple', 'average' and 'hard' categories. For each category, we applied specific strategy. This approach allowed us to outperform the existing state-of-art approaches by 3% on Terrorism domain and 6% on Management Succession domain. In our future work we plan to investigate the role of semantic relations and integrate ontology in the rule generation process. Another direction is to explore the use of bootstrapping and transduction approaches that may require less training instances.

References

H.L. Chieu and H.T. Ng. 2002. A Maximum Entropy Approach to Information Extraction from Semi-Structured and Free Text. *In Proc of AAAI-2002*, 786-791.

H. Cui, M.Y. Kan, and Chua T.S. 2005. *Generic Soft Pattern Models for Definitional Question Answering.* In Proc of ACM SIGIR-2005.

A. Culotta and J. Sorensen J. 2004. Dependency tree kernels for relation extraction. *In Proc of ACL-2004*.

F. Ciravegna. 2001. Adaptive Information Extraction from Text by Rule Induction and Generalization. *In Proc of IJCAI-2001*.

A. Dempster, N. Laird, and D. Rubin. 1977. *Maximum likelihood from incomplete data via the EM algorithm.* Journal of the Royal Statistical Society B, 39(1):1–38

K. Humphreys, G. Demetriou and R. Gaizuskas. 2000. Two applications of Information Extraction to Biological Science: Enzyme interactions and Protein structures. *In Proc of the Pacific Symposium on Biocomputing*, 502-513

M. Kaufmann. 1992. MUC-4. *In Proc of MUC-4.*

M. Kaufmann. 1995. MUC-6. *In Proc of MUC-6.*

J. Kim and D. Moldovan. 1995. Acquisition of linguistic patterns for knowledge-based information extraction. *IEEE Transactions on KDE,* 7(5): 713-724

D. Lin. 1997. Using Syntactic Dependency as Local Context to Resolve Word Sense Ambiguity. *In Proc of ACL-97.*

E. Riloff. 1996. Automatically Generating Extraction Patterns from Untagged Text. *In Proc of AAAI-96*, 1044-1049.

D. Roth and W. Yih. 2002. Probabilistic Reasoning for Entity & Relation Recognition. In Proc of *COLING-2002.*

S. Soderland, D. Fisher, J. Aseltine and W. Lehnert. 1995. Crystal: Inducing a Conceptual Dictionary. *In Proc of IJCAI-95*, 1314-1319.

S. Soderland. 1999. Learning Information Extraction Rules for Semi-Structured and Free Text. *Machine Learning* 34:233-272.

J. Xiao, T.S. Chua and H. Cui. 2004. Cascading Use of Soft and Hard Matching Pattern Rules for Weakly Supervised Information Extraction. *In Proc of COLING-2004.*

H. Yang, H. Cui, M.-Y. Kan, M. Maslennikov, L. Qiu and T.-S. Chua. 2003. QUALIFIER in TREC 12 QA Main Task. *In Proc of TREC-12,* 54-65.

R. Yangarber, W. Lin, R. Grishman. 2002. Unsupervised Learning of Generalized Names. *In Proc of COLING-2002.*

G.D. Zhou and J. Su. 2002. Named entity recognition using an HMM-based chunk tagger. *In Proc of ACL-2002,* 473-480