Using Predicate-Argument Structures for Information Extraction

Mihai Surdeanu and **Sanda Harabagiu** and **John Williams** and **Paul Aarseth**
Language Computer Corp.
Richardson, Texas 75080, USA
mihai,sanda@languagecomputer.com

Abstract

In this paper we present a novel, customizable IE paradigm that takes advantage of predicate-argument structures. We also introduce a new way of automatically identifying predicate argument structures, which is central to our IE paradigm. It is based on: (1) an extended set of features; and (2) inductive decision tree learning. The experimental results prove our claim that accurate predicate-argument structures enable high quality IE results.

1 Introduction

The goal of recent Information Extraction (IE) tasks was to provide event-level indexing into news stories, including news wire, radio and television sources. In this context, the purpose of the HUB Event-99 evaluations (Hirschman et al., 1999) was to capture information on some newsworthy classes of events, e.g. natural disasters, deaths, bombings, elections, financial fluctuations or illness outbreaks. The identification and selective extraction of relevant information is dictated by *templettes*. Event templettes are frame-like structures with slots representing the event basic information, such as main event participants, event outcome, time and location. For each type of event, a separate templette is defined. The slots fills consist of excerpts from text with pointers back into the original source material. Templettes are designed to support event-based browsing and search. Figure 1 illustrates a templette defined for "market changes" as well as the source of the slot fillers.

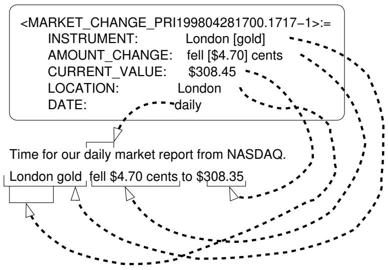

Figure 1: Templette filled with information about a market change event.

To date, some of the most successful IE techniques are built around a set of domain relevant linguistic patterns based on select verbs (e.g. *fall*, *gain* or *lose* for the "market change" topic). These patterns are matched against documents for identifying and extracting domain-relevant information. Such patterns are either handcrafted or acquired automatically. A rich literature covers methods of automatically acquiring IE patterns. Some of the most recent methods were reported in (Riloff, 1996; Yangarber et al., 2000).

To process texts efficiently and fast, domain patterns are ideally implemented as finite state automata (FSAs), a methodology pioneered in the FASTUS IE system (Hobbs et al., 1997). Although this paradigm is simple and elegant, it has the disadvantage that it is not easily portable from one domain of interest to the next.

In contrast, a new, truly domain-independent IE paradigm may be designed if we know (a) predicates relevant to a domain; and (b) which of their arguments fill templette slots. Central to this new way of extracting information from texts are systems that label predicate-argument structures on the output of full parsers. One such augmented parser, trained on data available from the PropBank project has been recently presented in (Gildea and Palmer, 2002). In this paper we describe a domain-independent IE paradigm that is based on predicate-argument structures identified automatically by two different methods: (1) the statistical method reported in (Gildea and Palmer, 2002); and (2) a new method based on inductive learning which obtains 17% higher F-score over the first method when tested on the same data. The accuracy enhancement of predicate argument recognition determines up to 14% better IE results. These results enforce our claim that predicate argument information for IE needs to be recognized with high accuracy.

The remainder of this paper is organized as follows. Section 2 reports on the parser that produces predicate-argument labels and compares it against the parser introduced in (Gildea and Palmer, 2002). Section 3 describes the pattern-free IE paradigm and compares it against FSA-based IE methods. Section 4 describes the integration of predicate-argument parsers into the IE paradigm and compares the results against a FSA-based IE system. Section 5 summarizes the conclusions.

2 Learning to Recognize Predicate-Argument Structures

2.1 The Data

Proposition Bank or PropBank is a one million word corpus annotated with predicate-argument structures. The corpus consists of the Penn Treebank 2 Wall Street Journal texts (*www.cis.upenn.edu/∼treebank*). The PropBank annotations, performed at University of Pennsylvania (*www.cis.upenn.edu/∼ace*) were described in (Kingsbury et al., 2002). To date PropBank has addressed only predicates lexicalized by verbs, proceeding from the most to the least common verbs while annotating verb predicates in the corpus. For any given predicate, a survey was made to determine the predicate usage and if required, the usages were divided in major senses. However, the senses are divided more on syntactic grounds than semantic, under the fundamental assumption that syntactic frames are direct reflections of underlying semantics.

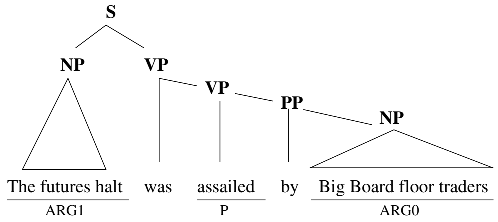

Figure 2: Sentence with annotated arguments

The set of syntactic frames are determined by diathesis alternations, as defined in (Levin, 1993). Each of these syntactic frames reflect underlying semantic components that constrain allowable arguments of predicates. The expected arguments of each predicate are numbered sequentially from Arg0 to Arg5. Regardless of the syntactic frame or verb sense, the arguments are similarly labeled to determine near-similarity of the predicates. The general procedure was to select for each verb the roles that seem to occur most frequently and use these roles as mnemonics for the predicate arguments. Generally, Arg0 would stand for *agent*, Arg1 for *direct object* or *theme* whereas Arg2 represents *indirect object*, *benefactive* or *instrument*, but mnemonics tend to be verb specific. For example, when retrieving the argument structure for the verb-predicate *assail* with the sense "to tear attack" from *www.cis.upenn.edu/∼cotton/cgi-bin/pblex_fmt.cgi*, we find Arg0:*agent*, Arg1:*entity assailed* and Arg2:*assailed for*. Additionally, the argument may include functional tags from Treebank, e.g. ArgM-DIR indicates a directional, ArgM-LOC indicates a locative, and ArgM-TMP stands for a temporal.

2.2 The Model

In previous work using the PropBank corpus, (Gildea and Palmer, 2002) proposed a model predicting argument roles using the same statistical method as the one employed by (Gildea and Jurafsky, 2002) for predicting semantic roles based on the FrameNet corpus (Baker et al., 1998). This statistical technique of labeling predicate argument operates on the output of the probabilistic parser reported in (Collins, 1997). It consists of two tasks: (1) identifying the parse tree constituents corresponding to arguments of each predicate encoded in PropBank; and (2) recognizing the role corresponding to each argument. Each task can be cast a separate classifier. For example, the result of the first classifier on the sentence illustrated in Figure 2 is the identification of the two NPs as arguments. The second classifier assigns the specific roles ARG1 and ARG0 given the predicate "assailed".

– PHRASE TYPE (pt): This feature indicates the syntactic type of the phrase labeled as a predicate argument, e.g. NP for ARG1 in Figure 2.
– PARSE TREE PATH (path): This feature contains the path in the parse tree between the predicate phrase and the argument phrase, expressed as a sequence of nonterminal labels linked by direction symbols (up or down), e.g. NP $\Uparrow$S $\Downarrow$ VP $\Downarrow$ VP for ARG1 in Figure 2.
– POSITION (pos) – Indicates if the constituent appears before or after the the predicate in the sentence.
– VOICE (voice) – This feature distinguishes between active or passive voice for the predicate phrase.
– HEAD WORD (hw) – This feature contains the head word of the evaluated phrase. Case and morphological information are preserved.
– GOVERNING CATEGORY (gov) – This feature applies to noun phrases only, and it indicates if the NP is dominated by a sentence phrase (typical for subject arguments with active−voice predicates), or by a verb phrase (typical for object arguments).
– PREDICATE WORD – In our implementation this feature consists of two components: (1) VERB: the word itself with the case and morphological information preserved; and (2) LEMMA which represents the verb normalized to lower case and infinitive form.

Figure 3: Feature Set 1

Statistical methods in general are hindered by the data sparsity problem. To achieve high accuracy and resolve the data sparsity problem the method reported in (Gildea and Palmer, 2002; Gildea and Jurafsky, 2002) employed a backoff solution based on a lattice that combines the model features. For practical reasons, this solution restricts the size of the feature sets. For example, the backoff lattice in (Gildea and Palmer, 2002) consists of eight connected nodes for a five-feature set. A larger set of features will determine a very complex backoff lattice. Consequently, no new intuitions may be tested as no new features can be easily added to the model.

In our studies we found that inductive learning through decision trees enabled us to easily test large sets of features and study the impact of each feature

– CONTENT WORD (cw) – Lexicalized feature that selects an informative word from the constituent, different from the head word.
PART OF SPEECH OF HEAD WORD (hPos) – The part of speech tag of the head word.
PART OF SPEECH OF CONTENT WORD (cPos) –The part of speech tag of the content word.
NAMED ENTITY CLASS OF CONTENT WORD (cNE) – The class of the named entity that includes the content word
BOOLEAN NAMED ENTITY FLAGS – A feature set comprising: – neOrganization: set to 1 if an organization is recognized in the phrase – neLocation: set to 1 a location is recognized in the phrase – nePerson: set to 1 if a person name is recognized in the phrase – neMoney: set to 1 if a currency expression is recognized in the phrase – nePercent: set to 1 if a percentage expression is recognized in the phrase – neTime: set to 1 if a time of day expression is recognized in the phrase – neDate: set to 1 if a date temporal expression is recognized in the phrase
PHRASAL VERB COLOCATIONS – Comprises two features: – pvcSum: the frequency with which a verb is immediately followed by any preposition or particle. – pvcMax: the frequency with which a verb is followed by its predominant preposition or particle.

Figure 4: Feature Set 2

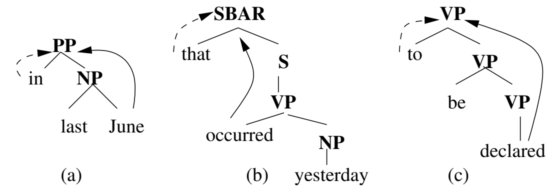

Figure 5: Sample phrases with the content word different than the head word. The head words are indicated by the dashed arrows. The content words are indicated by the continuous arrows.

on the augmented parser that outputs predicate argument structures. For this reason we used the C5 inductive decision tree learning algorithm (Quinlan, 2002) to implement both the classifier that identifies argument constituents and the classifier that labels arguments with their roles.

Our model considers two sets of features: Feature Set 1 (`FS1`): features used in the work reported in (Gildea and Palmer, 2002) and (Gildea and Jurafsky, 2002) ; and Feature Set 2 (`FS2`): a novel set of features introduced in this paper. `FS1` is illustrated in Figure 3 and `FS2` is illustrated in Figure 4.

In developing `FS2` we used the following observations:

Observation 1:
Because most of the predicate arguments are prepositional attachments (PP) or relative clauses (SBAR), often the head word (`hw`) feature from `FS1` is not in fact the most informative word in

H1: if phrase type is PP **then** select the right–most child Example: phrase = "in Texas", cw = "Texas"
H2: if phrase type is SBAR **then** select the left–most sentence (S*) clause Example: phrase = "that occurred yesterday", cw = "occurred"
H3: if phrase type is VP **then** **if** there is a VP child **then** select the left–most VP child **else** select the head word Example: phrase = "had placed", cw = "placed"
H4: if phrase type is ADVP **then** select the right–most child not IN or TO Example: phrase = "more than", cw = "more"
H5: if phrase type is ADJP **then** select the right–most adjective, verb, noun, or ADJP Example: phrase = "61 years old", cw = "old"
H6: for for all other phrase types **do** select the head word Example: phrase = "red house", cw = "house"

Figure 6: Heuristics for the detection of content words

the phrase. Figure 5 illustrates three examples of this situation. In Figure 5(a), the head word of the PP phrase is the preposition *in*, but *June* is at least as informative as the head word. Similarly, in Figure 5(b), the relative clause is featured only by the relative pronoun *that*, whereas the verb *occurred* should also be taken into account. Figure 5(c) shows another example of an infinitive verb phrase, in which the head word is *to*, whereas the verb *declared* should also be considered. Based on these observations, we introduced in `FS2` the CONTENT WORD (`cw`), which adds a new lexicalization from the argument constituent for better content representation. To select the content words we used the heuristics illustrated in Figure 6.

Observation 2:

After implementing `FS1`, we noticed that the `hw` feature was rarely used, and we believe that this happens because of data sparsity. The same was noticed for the `cw` feature from `FS2`. Therefore we decided to add two new features, namely the parts of speech of the head word and the content word respectively. These features are called `hPos` and `cPos` and are illustrated in Figure 4. Both these features generate an implicit yet simple backoff solution for the lexicalized features HEAD WORD (`hw`) and CONTENT WORD (`cw`).

Observation 3:

Predicate arguments often contain names or other expressions identified by named entity (NE) recognizers, e.g. dates, prices. Thus we believe that this form of semantic information should be introduced in the learning model. In `FS2` we added the following features: (a) the named entity class of the content word (`cNE`); and (b) a set of NE features that can take only Boolean values grouped as BOOLEAN NAMED ENTITY FEATURES and defined in Figure 4. The `cNE` feature helps recognize the argument roles, e.g. ARGM-LOC and ARGM-TMP, when location or temporal expressions are identified. The Boolean NE flags provide information useful in processing complex nominals occurring in argument constituents. For example, in Figure 2 ARG0 is featured not only by the word *traders* but also by ORGANIZATION, the semantic class of the name *Big Board*.

Observation 4:

Predicate argument structures are recognized accurately when both predicates and arguments are correctly identified. Often, predicates are lexicalized by phrasal verbs, e.g. *put up*, *put off*. To identify correctly the verb particle and capture it in the structure of predicates instead of the argument structure, we introduced two collocation features that measure the frequency with which verbs and succeeding prepositions cooccurr in the corpus. The features are pvcSum and pvcMax and are defined in Figure 4.

2.3 The Experiments

The results presented in this paper were obtained by training on Proposition Bank (PropBank) release 2002/7/15 (Kingsbury et al., 2002). Syntactic information was extracted from the gold-standard parses in TreeBank Release 2. As named entity information is not available in PropBank/TreeBank we tagged the training corpus with NE information using an open-domain NE recognizer, having 96% F-measure on the MUC6[1] data. We reserved section 23 of PropBank/TreeBank for testing, and we trained on the rest. Due to memory limitations on our hardware, for the argument finding task we trained on the first 150 KB of TreeBank (about 11% of TreeBank), and

[1]The Message Understanding Conferences (MUC) were IE evaluation exercises in the 90s. Starting with MUC6 named entity data was available.

for the role assignment task on the first 75 KB of argument constituents (about 60% of PropBank annotations).

Table 1 shows the results obtained by our inductive learning approach. The first column describes the feature sets used in each of the 7 experiments performed. The following three columns indicate the precision (P), recall (R), and F-measure (F_1)[2] obtained for the task of identifying argument constituents. The last column shows the accuracy (A) for the role assignment task using known argument constituents. The first row in Table 1 lists the results obtained when using only the `FS1` features. The next five lines list the individual contributions of each of the newly added features when combined with the `FS1` features. The last line shows the results obtained when all features from `FS1` and `FS2` were used.

Table 1 shows that the new features increase the argument identification F-measure by 3.61%, and the role assignment accuracy with 4.29%. For the argument identification task, the head and content word features have a significant contribution for the task precision, whereas NE features contribute significantly to the task recall. For the role assignment task the best features from the feature set `FS2` are the content word features (`cw` and `cPos`) and the Boolean NE flags, which show that semantic information, even if minimal, is important for role classification. Surprisingly, the phrasal verb collocation features did not help for any of the tasks, but they were useful for boosting the decision trees. Decision tree learning provided by C5 (Quinlan, 2002) has built in support for boosting. We used it and obtained improvements for both tasks. The best F-measure obtained for argument constituent identification was 88.98% in the fifth iteration (a 0.76% improvement). The best accuracy for role assignment was 83.74% in the eight iteration (a 0.69% improvement)[3]. We further analyzed the boosted trees and noticed that phrasal verb collocation features were mainly responsible for the improvements. This is the rationale for including them in the `FS2` set.

We also were interested in comparing the results

[2] $F_1 = \frac{2P \times R}{P+R}$

[3] These results, listed also on the last line of Table 2, differ from those in Table 1 because they were produced after the boosting took place.

Features	Arg P	Arg R	Arg F_1	Role A
`FS1`	84.96	84.26	84.61	78.76
`FS1` + `hPos`	92.24	84.50	88.20	79.04
`FS1` + `cw`, `cPos`	92.19	84.67	88.27	80.80
`FS1` + `cNE`	83.93	85.69	84.80	79.85
`FS1` + NE flags	87.78	85.71	86.73	81.28
`FS1` + pvcSum + pvcMax	84.88	82.77	83.81	78.62
`FS1` + `FS2`	91.62	85.06	**88.22**	**83.05**

Table 1: Inductive learning results for argument identification and role assignment

Model	Implementation	Arg F_1	Role A
Statistical	(Gildea and Palmer)	-	82.8
	This study	71.86	78.87
Decision Trees	`FS1`	84.61	78.76
	`FS1` + `FS2`	88.98	83.74

Table 2: Comparison of statistical and decision tree learning models

of the decision-tree-based method against the results obtained by the statistical approach reported in (Gildea and Palmer, 2002). Table 2 summarizes the results. (Gildea and Palmer, 2002) report the results listed on the first line of Table 2. Because no F-scores were reported for the argument identification task, we re-implemented the model and obtained the results listed on the second line. It looks like we had some implementation differences, and our results for the argument role classification task were slightly worse. However, we used our results for the statistical model for comparing with the inductive learning model because we used the same feature extraction code for both models. Lines 3 and 4 list the results of the inductive learning model with boosting enabled, when the features were only from `FS1`, and from `FS1` and `FS2` respectively. When comparing the results obtained for both models when using only features from `FS1`, we find that almost the same results were obtained for role classification, but an enhancement of almost 13% was obtained when recognizing argument constituents. When comparing the statistical model with the inductive model that uses all features, there is an enhancement of 17.12% for argument identification and 4.87% for argument role recognition.

Another significant advantage of our inductive learning approach is that it scales better to un-

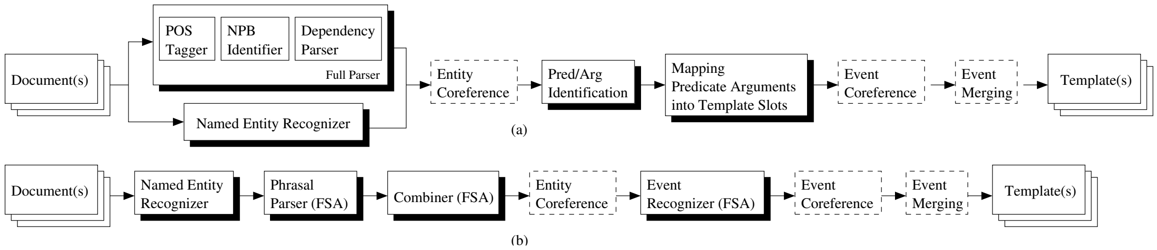

Figure 7: IE architectures: (a) Architecture based on predicate/argument relations; (b) FSA-based IE system

known predicates. The statistical model introduced in Gildea and Jurafsky (2002) uses predicate lexical information at most levels in the probability lattice, hence its scalability to unknown predicates is limited. In contrast, the decision tree approach uses predicate lexical information only for 5% of the branching decisions recorded when testing the role assignment task, and only for 0.01% of the branching decisions seen during the argument constituent identification evaluation.

3 The IE Paradigm

Figure 7(a) illustrates an IE architecture that employs predicate argument structures. Documents are processed in parallel to: (1) parse them syntactically, and (2) recognize the NEs. The full parser first performs part-of-speech (POS) tagging using transformation based learning (TBL) (Brill, 1995). Then non-recursive, or basic, noun phrases (NPB) are identified using the TBL method reported in (Ngai and Florian, 2001). At last, the dependency parser presented in (Collins, 1997) is used to generate the full parse. This approach allows us to parse the sentences with less than 40 words from TreeBank section 23 with an F-measure slightly over 85% at an average of 0.12 seconds/sentence on a 2GHz Pentium IV computer.

The parse texts marked with NE tags are passed to a module that identifies entity coreference in documents, resolving pronominal and nominal anaphors and normalizing coreferring expressions. The parses are also used by a module that recognizes predicate argument structures with any of the methods described in Section 2.

For each templette modeling a different domain a mapping between predicate arguments and templette slots is produced. Figure 8 illustrates the mapping produced for two Event99 domains. The *"market change"* domain monitors changes (AMOUNT_CHANGE) and current values (CURRENT_VALUE) for financial instruments (INSTRUMENT). The *"death"* domain extracts the description of the person deceased (DECEASED), the manner of death (MANNER_OF_DEATH), and, if applicable, the person to whom the death is attributed (AGENT_OF_DEATH).

ARG1 and MARKET_CHANGE_VERB ⟶ INSTRUMENT
ARG2 and (MONEY or PERCENT or NUMBER or QUANTITY) and MARKET_CHANGE_VERB ⟶ AMOUNT_CHANGE
(ARG4 or ARGM_DIR) and NUMBER and MARKET_CHANGE_VERB ⟶ CURRENT_VALUE (a)

(PERSON and ARG0 and DIE_VERB) or (PERSON and ARG1 and KILL_VERB) ⟶ DECEASED
(ARG0 and KILL_VERB) or (ARG1 and DIE_VERB) ⟶ AGENT_OF_DEATH
(ARGM−TMP and ILNESS_NOUN) or KILL_VERB or DIE_VERB ⟶ MANNER_OF_DEATH
ARGM−TMP and DATE ⟶ DATE
(ARGM−LOC or ARGM−TMP) and LOCATION ⟶ LOCATION (b)

Figure 8: Mapping rules between predicate arguments and templette slots for: (a) the "market change" domain, and (b) the "death" domain

To produce the mappings we used training data that consists of: (1) texts, and (2) their corresponding filled templettes. Each templette has pointers back to the source text similarly to the example presented in Figure 1. When the predicate argument structures were identified, the mappings were collected as illustrated in Figure 9. Figure 9(a) shows an interesting aspect of the mappings. Although the role classification of the last argument is incorrect (it should have been identified as ARG4), it is mapped into the CURRENT-VALUE slot. This shows how the mappings resolve incorrect but consistent classifications. Figure 9(b) shows the flexibility of the system to identify and classify constituents that are not close to the predicate phrase (ARG0). This is a clear ad-

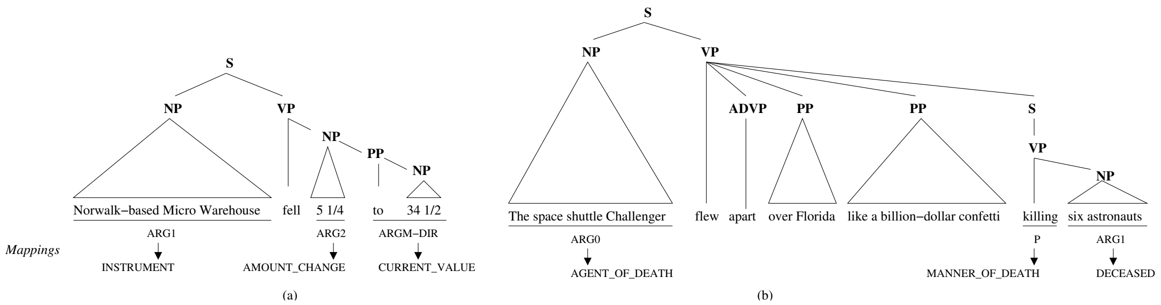

Figure 9: Predicate argument mapping examples for: (a) the "market change" domain, and (b) the "death" domain

vantage over the FSA-based system, which in fact missed the AGENT-OF-DEATH in this sentence. Because several templettes might describe the same event, event coreference is processed and, based on the results, templettes are merged when necessary.

The IE architecture in Figure 7(a) may be compared with the IE architecture with cascaded FSA represented in Figure 7(b) and reported in (Surdeanu and Harabagiu, 2002). Both architectures share the same NER, coreference and merging modules. Specific to the FSA-based architecture are the phrasal parser, which identifies simple phrases such as basic noun or verb phrases (some of them domain specific), the combiner, which builds domain-dependent complex phrases, and the event recognizer, which detects the domain-specific Subject-Verb-Object (SVO) patterns. An example of a pattern used by the FSA-based architecture is: [DEATH-CAUSE KILL-VERB PERSON], where DEATH-CAUSE may identify more than 20 lexemes, e.g. *wreck*, *catastrophe*, *malpractice*, and more than 20 verbs are KILL-VERBS, e.g. *murder*, *execute*, *behead*, *slay*. Most importantly, each pattern must recognize up to 26 syntactic variations, e.g. determined by the active or passive form of the verb, relative subjects or objects etc. Predicate argument structures offer the great advantage that syntactic variations do not need to be accounted by IE systems anymore.

Because entity and event coreference, as well as templette merging will attempt to recover from partial patterns or predicate argument recognitions, and our goal is to compare the usage of FSA patterns versus predicate argument structures, we decided to disable the coreference and merging modules. This explains why in Figure 7 these modules are represented with dashed lines.

System	Market Change	Death
Pred/Args Statistical	68.9%	58.4%
Pred/Args Inductive	82.8%	67.0%
FSA	91.3%	72.7%

Table 3: Templette F-measure (F_1) scores for the two domains investigated

System	Correct	Missed	Incorrect
Pred/Args Statistical	26	16	3
Pred/Args Inductive	33	9	2
FSA	38	4	2

Table 4: Number of event structures (FSA patterns or predicate argument structures) matched

4 Experiments with The Integration of Predicate Argument Structures in IE

To evaluate the proposed IE paradigm we selected two Event99 domains: *"market change"*, which tracks changes in stock indexes, and *"death"*, which extracts all manners of human deaths. These domains were selected because most of the domain information can be processed without needing entity or event coreference. Moreover, one of the domains (market change) uses verbs commonly used in PropBank/TreeBank, while the other (death) uses relatively unknown verbs, so we can also evaluate how well the system scales to verbs unseen in training.

Table 3 lists the F-scores for the two domains. The first line of the Table lists the results obtained by the IE architecture illustrated in Figure 7(a) when the predicate argument structures were identified by the statistical model. The next line shows the same results for the inductive learning model. The last line shows the results for the IE architecture in Figure 7(b). The results obtained by the FSA-based IE were the best, but they were made possible by hand-crafted patterns requiring an effort of 10 person days per domain. The only human effort necessary in the new IE paradigm was imposed by the generation of mappings between arguments and templette slots, accomplished in less than 2 hours per domain, given that the training templettes are known. Additionally, it is easier to automatically learn these mappings than to acquire FSA patterns.

Table 3 also shows that the new IE paradigm performs better when the predicate argument structures are recognized with the inductive learning model. The cause is the substantial difference in quality of the argument identification task between the two models. The Table shows that the new IE paradigm with the inductive learning model achieves about 90% of the performance of the FSA-based system for both domains, even though one of the domains uses mainly verbs rarely seen in training (e.g. "die" appears 5 times in PropBank).

Another way of evaluating the integration of predicate argument structures in IE is by comparing the number of events identified by each architecture. Table 4 shows the results. Once again, the new IE paradigm performs better when the predicate argument structures are recognized with the inductive learning model. More events are missed by the statistical model which does not recognize argument constituents as well the inductive learning model.

5 Conclusion

This paper reports on a novel inductive learning method for identifying predicate argument structures in text. The proposed approach achieves over 88% F-measure for the problem of identifying argument constituents, and over 83% accuracy for the task of assigning roles to pre-identified argument constituents. Because predicate lexical information is used for less than 5% of the branching decisions, the generated classifier scales better than the statistical method from (Gildea and Palmer, 2002) to unknown predicates. This way of identifying predicate argument structures is a central piece of an IE paradigm easily customizable to new domains. The performance degradation of this paradigm when compared to IE systems based on hand-crafted patterns is only 10%.

References

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet Project. In *Proceedings of COLING/ACL '98*:86-90,. Montreal, Canada.

Eric Brill. 1995. Transformation-Based Error-Driven Learning and Natural Language Processing: A Case Study in Part of Speech Tagging. *Computational Linguistics*.

Michael Collins. 1997. Three Generative, Lexicalized Models for Statistical Parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL 1997)*:16-23, Madrid, Spain.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic Labeling of Semantic Roles. *Computational Linguistics*, 28(3):245-288.

Daniel Gildea and Martha Palmer. 2002. The Necessity of Parsing for Predicate Argument Recognition. In *Proceedings of the 40th Meeting of the Association for Computational Linguistics (ACL 2002)*:239-246, Philadelphia, PA.

Lynette Hirschman, Patricia Robinson, Lisa Ferro, Nancy Chinchor, Erica Brown, Ralph Grishman, Beth Sundheim 1999. Hub-4 Event99 General Guidelines and Templettes.

Jerry R. Hobbs, Douglas Appelt, John Bear, David Israel, Megumi Kameyama, Mark E. Stickel, and Mabry Tyson. 1997. FASTUS: A Cascaded Finite-State Transducer for Extracting Information from Natural-Language Text. In *Finite-State Language Processing*, pages 383-406, MIT Press, Cambridge, MA.

Paul Kingsbury, Martha Palmer, and Mitch Marcus. 2002. Adding Semantic Annotation to the Penn TreeBank. In *Proceedings of the Human Language Technology Conference (HLT 2002)*:252-256, San Diego, California.

Beth Levin. 1993. English Verb Classes and Alternations a Preliminary Investigation. University of Chicago Press.

Grace Ngai and Radu Florian. 2001. Transformation-Based Learning in The Fast Lane. In *Proceedings of the North American Association for Computational Linguistics (NAACL 2001)*:40-47.

Ross Quinlan. 2002. Data Mining Tools See5 and C5.0. http://www.rulequest.com/see5-info.html.

Ellen Riloff and Rosie Jones. 1996. Automatically Generating Extraction Patterns from Untagged Text. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96))*:1044-1049.

Mihai Surdeanu and Sanda Harabagiu. 2002. Infrastructure for Open-Domain Information Extraction In *Proceedings of the Human Language Technology Conference (HLT 2002)*:325-330.

Roman Yangarber, Ralph Grishman, Pasi Tapainen and Silja Huttunen, 2000. Automatic Acquisition of Domain Knowledge for Information Extraction. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING-2000)*: 940-946, Saarbrucken, Germany.