Template-Based Information Extraction without the Templates

Nathanael Chambers and **Dan Jurafsky**
Department of Computer Science
Stanford University
{natec,jurafsky}@stanford.edu

Abstract

Standard algorithms for template-based information extraction (IE) require predefined template schemas, and often labeled data, to learn to extract their slot fillers (e.g., an *embassy* is the *Target* of a *Bombing* template). This paper describes an approach to template-based IE that removes this requirement and performs extraction without knowing the template structure in advance. Our algorithm instead learns the template structure automatically from raw text, inducing template schemas as sets of linked events (e.g., bombings include *detonate*, *set off*, and *destroy* events) associated with semantic roles. We also solve the standard IE task, using the induced syntactic patterns to extract role fillers from specific documents. We evaluate on the MUC-4 terrorism dataset and show that we induce template structure very similar to hand-created gold structure, and we extract role fillers with an F1 score of .40, approaching the performance of algorithms that require full knowledge of the templates.

1 Introduction

A *template* defines a specific type of event (e.g., a bombing) with a set of *semantic roles* (or slots) for the typical entities involved in such an event (e.g., perpetrator, target, instrument). In contrast to work in relation discovery that focuses on learning atomic facts (Banko et al., 2007a; Carlson et al., 2010), templates can extract a richer representation of a particular domain. However, unlike relation discovery, most template-based IE approaches assume foreknowledge of the domain's templates. Very little work addresses how to learn the template structure itself. Our goal in this paper is to perform the standard template filling task, but to first automatically induce the templates from an unlabeled corpus.

There are many ways to represent events, ranging from role-based representations such as frames (Baker et al., 1998) to sequential events in scripts (Schank and Abelson, 1977) and narrative schemas (Chambers and Jurafsky, 2009; Kasch and Oates, 2010). Our approach learns narrative-like knowledge in the form of IE templates; we learn sets of related events and semantic roles, as shown in this sample output from our system:

Bombing Template
{detonate, blow up, plant, explode, defuse, destroy}

Perpetrator:	Person who detonates, plants, blows up
Instrument:	Object that is planted, detonated, defused
Target:	Object that is destroyed, is blown up

A semantic role, such as *target*, is a cluster of syntactic functions of the template's event words (e.g., the objects of *detonate* and *explode*). Our goal is to characterize a domain by learning this template structure completely automatically. We learn templates by first clustering event words based on their proximity in a training corpus. We then use a novel approach to role induction that clusters the syntactic functions of these events based on selectional preferences and coreferring arguments. The induced roles are template-specific (e.g., perpetrator), not universal (e.g., agent or patient) or verb-specific.

After learning a domain's template schemas, we perform the standard IE task of role filling from individual documents, for example:

Perpetrator:	guerrillas
Instrument:	dynamite
Target:	embassy

This extraction stage identifies entities using the learned syntactic functions of our roles. We evaluate on the MUC-4 terrorism corpus with results approaching those of supervised systems.

The core of this paper focuses on how to characterize a domain-specific corpus by learning rich template structure. We describe how to first expand the small corpus' size, how to cluster its events, and finally how to induce semantic roles. Section 5 then describes the extraction algorithm, followed by evaluations against previous work in section 6 and 7.

2 Previous Work

Many template extraction algorithms require full knowledge of the templates and labeled corpora, such as in rule-based systems (Chinchor et al., 1993; Rau et al., 1992) and modern supervised classifiers (Freitag, 1998; Chieu et al., 2003; Bunescu and Mooney, 2004; Patwardhan and Riloff, 2009). Classifiers rely on the labeled examples' surrounding context for features such as nearby tokens, document position, syntax, named entities, semantic classes, and discourse relations (Maslennikov and Chua, 2007). Ji and Grishman (2008) also supplemented labeled with unlabeled data.

Weakly supervised approaches remove some of the need for fully labeled data. Most still require the templates and their slots. One common approach is to begin with unlabeled, but clustered event-specific documents, and extract common word patterns as extractors (Riloff and Schmelzenbach, 1998; Sudo et al., 2003; Riloff et al., 2005; Patwardhan and Riloff, 2007). Filatova et al. (2006) integrate named entities into pattern learning (*PERSON won*) to approximate unknown semantic roles. Bootstrapping with seed examples of known slot fillers has been shown to be effective (Surdeanu et al., 2006; Yangarber et al., 2000). In contrast, this paper removes these data assumptions, learning instead from a corpus of unknown events and unclustered documents, without seed examples.

Shinyama and Sekine (2006) describe an approach to *template learning* without labeled data. They present *unrestricted relation discovery* as a means of discovering relations in unlabeled documents, and extract their fillers. Central to the algorithm is collecting multiple documents describing the same exact event (e.g. Hurricane Ivan), and observing repeated word patterns across documents connecting the same proper nouns. Learned patterns represent binary relations, and they show how to construct tables of extracted entities for these relations. Our approach draws on this idea of using unlabeled documents to discover relations in text, and of defining semantic roles by sets of entities. However, the limitations to their approach are that (1) redundant documents about specific events are required, (2) relations are binary, and (3) only slots with named entities are learned. We will extend their work by showing how to learn without these assumptions, obviating the need for redundant documents, and learning templates with any type and any number of slots.

Large-scale learning of scripts and narrative schemas also captures template-like knowledge from unlabeled text (Chambers and Jurafsky, 2008; Kasch and Oates, 2010). Scripts are sets of related event words and semantic roles learned by linking syntactic functions with coreferring arguments. While they learn interesting event structure, the structures are limited to frequent topics in a large corpus. We borrow ideas from this work as well, but our goal is to instead characterize a specific domain with limited data. Further, we are the first to apply this knowledge to the IE task of filling in template mentions in documents.

In summary, our work extends previous work on unsupervised IE in a number of ways. We are the first to learn MUC-4 templates, and we are the first to extract entities without knowing how many templates exist, without examples of slot fillers, and without event-clustered documents.

3 The Domain and its Templates

Our goal is to learn the general event structure of a domain, and then extract the instances of each learned event. In order to measure performance in both tasks (learning structure and extracting instances), we use the terrorism corpus of MUC-4 (Sundheim, 1991) as our target domain. This corpus was chosen because it is annotated with templates that describe all of the entities involved in each event. An example snippet from a *bombing* document is given here:

*The terrorists **used** explosives against the town hall. El Comercio reported that alleged Shining Path members also **attacked** public facilities in huarpacha, Ambo, tomayquichua, and kichki. Municipal official Sergio Horna was seriously **wounded** in an explosion in Ambo.*

The entities from this document fill the following slots in a MUC-4 bombing template.

Perp: Shining Path members	**Victim**: Sergio Horna
Target: public facilities	**Instrument**: explosives

We focus on these four string-based slots[1] from the MUC-4 corpus, as is standard in this task. The corpus consists of 1300 documents, 733 of which are labeled with at least one template. There are six types of templates, but only four are modestly frequent: *bombing* (208 docs), *kidnap* (83 docs), *attack* (479 docs), and *arson* (40 docs). 567 documents do not have any templates. Our learning algorithm does not know which documents contain (or do not contain) which templates. After learning event words that represent templates, we induce their slots, not knowing a priori how many there are, and then fill them in by extracting entities as in the standard task. In our example above, the three bold verbs (use, attack, wound) indicate the Bombing template, and their syntactic arguments fill its slots.

4 Learning Templates from Raw Text

Our goal is to learn templates that characterize a domain as described in unclustered, unlabeled documents. This presents a two-fold problem to the learner: it does not know how many events exist, and it does not know which documents describe which event (some may describe multiple events). We approach this problem with a three step process: (1) cluster the domain's event patterns to approximate the template topics, (2) build a new corpus *specific to each cluster* by retrieving documents from a larger unrelated corpus, (3) induce each template's slots using its new (larger) corpus of documents.

4.1 Clustering Events to Learn Templates

We cluster *event patterns* to create templates. An *event pattern* is either (1) a verb, (2) a noun in WordNet under the Event synset, or (3) a verb and the head word of its syntactic object. Examples of each include (1) 'explode', (2) 'explosion', and (3) 'explode:bomb'. We also tag the corpus with an NER system and allow patterns to include named entity types, e.g., 'kidnap:PERSON'. These patterns are crucially needed later to learn a template's slots. However, we first need an algorithm to cluster these patterns to learn the domain's core events. We consider two unsupervised algorithms: Latent Dirichlet Allocation (LDA) (Blei et al., 2003), and agglomerative clustering based on word distance.

4.1.1 LDA for Unknown Data

LDA is a probabilistic model that treats documents as mixtures of topics. It learns topics as discrete distributions (multinomials) over the event patterns, and thus meets our needs as it clusters patterns based on co-occurrence in documents. The algorithm requires the number of topics to be known ahead of time, but in practice this number is set relatively high and the resulting topics are still useful. Our best performing LDA model used 200 topics. We had mixed success with LDA though, and ultimately found our next approach performed slightly better on the document classification evaluation.

4.1.2 Clustering on Event Distance

Agglomerative clustering does not require foreknowledge of the templates, but its success relies on how event pattern similarity is determined.

Ideally, we want to learn that *detonate* and *destroy* belong in the same cluster representing a bombing. Vector-based approaches are often adopted to represent words as feature vectors and compute their distance with cosine similarity. Unfortunately, these approaches typically learn clusters of *synonymous* words that can miss detonate and destroy. Our goal is to instead capture world knowledge of co-occuring events. We thus adopt an assumption that *closeness* in the world is reflected by *closeness* in a text's discourse. We hypothesize that two patterns are related if they occur near each other in a document more often than chance.

Let $g(w_i, w_j)$ be the distance between two events (1 if in the same sentence, 2 in neighboring, etc). Let $C_{dist}(w_i, w_j)$ be the distance-weighted frequency of

[1]There are two Perpetrator slots in MUC-4: Organization and Individual. We consider their union as a single slot.

kidnap: kidnap, kidnap:PER, abduct, release, kidnapping, ransom, robbery, registration
bombing: explode, blow up, locate, place:bomb, detonate, damage, explosion, cause, damage, ...
attack: kill, shoot down, down, kill:civilian, kill:PER, kill:soldier, kill:member, killing, shoot:PER, wave, ...
arson: burn, search, burning, clip, collaborate, ...

Figure 1: The 4 clusters mapped to MUC-4 templates.

two events occurring together:

$$C_{dist}(w_i, w_j) = \sum_{d \in D} \sum_{w_i, w_j \in d} 1 - log_4(g(w_i, w_j)) \quad (1)$$

where d is a document in the set of all documents D. The base 4 logarithm discounts neighboring sentences by 0.5 and within the same sentence scores 1. Using this definition of distance, pointwise mutual information measures our similarity of two events:

$$pmi(w_i, w_j) = P_{dist}(w_i, w_j)/(P(w_i)P(w_j)) \quad (2)$$

$$P(w_i) = \frac{C(w_i)}{\sum_j C(w_j)} \quad (3)$$

$$P_{dist}(w_i, w_j) = \frac{C_{dist}(w_i, w_j)}{\sum_k \sum_l C_{dist}(w_k, w_l)} \quad (4)$$

We run agglomerative clustering with pmi over all event patterns. Merging decisions use the average link score between all new links across two clusters. As with all clustering algorithms, a stopping criterion is needed. We continue merging clusters until any single cluster grows beyond m patterns. We briefly inspected the clustering process and chose $m = 40$ to prevent learned scenarios from intuitively growing too large and ambiguous. Post-evaluation analysis shows that this value has wide flexibility. For example, the Kidnap and Arson clusters are unchanged in $30 < m < 80$, and Bombing unchanged in $30 < m < 50$. Figure 1 shows 3 clusters (of 77 learned) that characterize the main template types.

4.2 Information Retrieval for Templates

Learning a domain often suffers from a lack of training data. The previous section clustered events from the MUC-4 corpus, but its 1300 documents do not provide enough examples of verbs and argument counts to further learn the semantic roles in each cluster. Our solution is to assemble a larger *IR-corpus* of documents for each cluster. For example, MUC-4 labels 83 documents with Kidnap, but our learned cluster (*kidnap*, *abduct*, *release*, ...) retrieved 3954 documents from a general corpus.

We use the Associated Press and New York Times sections of the Gigaword Corpus (Graff, 2002) as our general corpus. These sections include approximately 3.5 million news articles spanning 12 years.

Our retrieval algorithm retrieves documents that score highly with a cluster's tokens. The document score is defined by two common metrics: word match, and word coverage. A document's match score is defined as the average number of times the words in cluster c appear in document d:

$$avgm(d, c) = \frac{\sum_{w \in c} \sum_{t \in d} 1\{w = t\}}{|c|} \quad (5)$$

We define *word coverage* as the number of seen cluster words. Coverage penalizes documents that score highly by repeating a single cluster word a lot. We only score a document if its coverage, $cvg(d, c)$, is at least 3 words (or less for tiny clusters):

$$ir(d, c) = \begin{cases} avgm(d, c) & \text{if } cvg(d, c) > min(3, |c|/4) \\ 0 & \text{otherwise} \end{cases}$$

A document d is retrieved for a cluster c if $ir(d, c) > 0.4$. Finally, we emphasize precision by pruning away 50% of a cluster's retrieved documents that are farthest in distance from the mean document of the retrieved set. Distance is the cosine similarity between bag-of-words vector representations. The confidence value of 0.4 was chosen from a manual inspection among a single cluster's retrieved documents. Pruning 50% was arbitrarily chosen to improve precision, and we did not experiment with other quantities. A search for optimum parameter values may lead to better results.

4.3 Inducing Semantic Roles (Slots)

Having successfully clustered event words and retrieved an *IR-corpus* for each cluster, we now address the problem of *inducing semantic roles*. Our learned roles will then extract entities in the next section and we will evaluate their per-role accuracy.

Most work on unsupervised role induction focuses on learning *verb-specific* roles, starting with seed examples (Swier and Stevenson, 2004; He and

Gildea, 2006) and/or knowing the number of roles (Grenager and Manning, 2006; Lang and Lapata, 2010). Our previous work (Chambers and Jurafsky, 2009) learned *situation-specific* roles over narrative schemas, similar to frame roles in FrameNet (Baker et al., 1998). Schemas link the syntactic relations of verbs by clustering them based on observing coreferring arguments in those positions. This paper extends this intuition by introducing a new vector-based approach to coreference similarity.

4.3.1 Syntactic Relations as Roles

We learn the roles of cluster C by clustering the syntactic relations R_C of its words. Consider the following example:

C = {*go off, explode, set off, damage, destroy*}
R_C = {*go_off:s, go_off:p_in, explode:s, set_off:s*}

where *verb:s* is the verb's subject, *:o* the object, and *p_in* a preposition. We ideally want to cluster R_C as:

bomb = {*go_off:s, explode:s, set_off:o, destroy:s*}
suspect = {*set_off:s*}
target = {*go_off:p_in, destroy:o*}

We want to cluster all subjects, objects, and prepositions. Passive voice is normalized to active[2].

We adopt two views of relation similarity: coreferring arguments and selectional preferences. Chambers and Jurafsky (2008) observed that coreferring arguments suggest a semantic relation between two predicates. In the sentence, *he ran and then he fell*, the subjects of run and fall corefer, and so they likely belong to the same scenario-specific semantic role. We applied this idea to a new vector similarity framework. We represent a relation as a vector of all relations with which their arguments coreferred. For instance, arguments of the relation *go_off:s* were seen coreferring with mentions in *plant:o*, *set_off:o* and *injure:s*. We represent *go_off:s* as a vector of these relation counts, calling this its *coref vector representation*.

Selectional preferences (SPs) are also useful in measuring similarity (Erk and Pado, 2008). A relation can be represented as a vector of its observed arguments during training. The SPs for *go_off:s* in our data include {*bomb, device, charge, explosion*}.

We measure similarity using cosine similarity between the vectors in both approaches. However, coreference and SPs measure different types of similarity. Coreference is a looser narrative similarity (bombings cause injuries), while SPs capture synonymy (plant and place have similar arguments). We observed that many narrative relations are not synonymous, and vice versa. We thus take the maximum of either cosine score as our final similarity metric between two relations. We then back off to the average of the two cosine scores if the max is not confident (less than 0.7); the average penalizes the pair. We chose the value of 0.7 from a grid search to optimize extraction results on the training set.

4.3.2 Clustering Syntactic Functions

We use agglomerative clustering with the above pairwise similarity metric. Cluster similarity is the average link score over all new links crossing two clusters. We include the following sparsity penalty $r(c_a, c_b)$ if there are too few links between clusters c_a and c_b.

$$score(c_a, c_b) = \sum_{w_i \in c_a} \sum_{w_j \in c_b} sim(w_i, w_j) * r(c_a, c_b) \quad (6)$$

$$r(c_a, c_b) = \frac{\sum_{w_i \in c_a} \sum_{w_j \in c_b} 1\{sim(w_i, w_j) > 0\}}{\sum_{w_i \in c_a} \sum_{w_j \in c_b} 1} \quad (7)$$

This penalizes clusters from merging when they share only a few high scoring edges. Clustering stops when the merged cluster scores drop below a threshold optimized to extraction performance on the training data.

We also begin with two assumptions about syntactic functions and semantic roles. The first assumes that the subject and object of a verb carry different semantic roles. For instance, the subject of *sell* fills a different role (Seller) than the object (Good). The second assumption is that each semantic role has a high-level entity type. For instance, the subject of *sell* is a Person or Organization, and the object is a Physical Object.

We implement the first assumption as a constraint in the clustering algorithm, preventing two clusters from merging if their union contains the same verb's subject and object.

We implement the second assumption by automatically labeling each syntactic function with a role type based on its observed arguments. The role types are broad general classes: *Person/Org*, *Physical Object*, or *Other*. A syntactic function is labeled as a

[2]We use the Stanford Parser at nlp.stanford.edu/software

Bombing Template (MUC-4)

Perpetrator *Person/Org* who detonates, blows up, plants, hurls, stages, is detained, is suspected, is blamed on, launches

Instrument A *physical object* that is exploded, explodes, is hurled, causes, goes off, is planted, damages, is set off, is defused

Target A *physical object* that is damaged, is destroyed, is exploded at, is damaged, is thrown at, is hit, is struck

Police *Person/Org* who raids, questions, discovers, investigates, defuses, arrests

N/A A *physical object* that is blown up, destroys

Attack/Shooting Template (MUC-4)

Perpetrator *Person/Org* who assassinates, patrols, ambushes, raids, shoots, is linked to

Victim *Person/Org* who is assassinated, is toppled, is gunned down, is executed, is evacuated

Target *Person/Org* who is hit, is struck, is downed, is set fire to, is blown up, surrounded

Instrument A *physical object* that is fired, injures, downs, is set off, is exploded

Kidnap Template (MUC-4)

Perpetrator *Person/Org* who releases, abducts, kidnaps, ambushes, holds, forces, captures, is imprisoned, frees

Target *Person/Org* who is kidnapped, is released, is freed, escapes, disappears, travels, is harmed, is threatened

Police *Person/Org* who rules out, negotiates, condemns, is pressured, finds, arrests, combs

Weapons Smuggling Template (NEW)

Perpetrator *Person/Org* who smuggles, is seized from, is captured, is detained

Police *Person/Org* who raids, seizes, captures, confiscates, detains, investigates

Instrument A *physical object* that is smuggled, is seized, is confiscated, is transported

Election Template (NEW)

Voter *Person/Org* who chooses, is intimidated, favors, is appealed to, turns out

Government *Person/Org* who authorizes, is chosen, blames, authorizes, denies

Candidate *Person/Org* who resigns, unites, advocates, manipulates, pledges, is blamed

Figure 2: Five learned example templates. All knowledge except the template/role names (e.g., 'Victim') is learned.

class if 20% of its arguments appear under the corresponding WordNet synset[3], or if the NER system labels them as such. Once labeled by type, we separately cluster the syntactic functions for each role type. For instance, Person functions are clustered separate from Physical Object functions. Figure 2 shows some of the resulting roles.

Finally, since agglomerative clustering makes hard decisions, related events to a template may have been excluded in the initial event clustering stage. To address this problem, we identify the 200 *nearby events* to each event cluster. These are simply the top scoring event patterns with the cluster's original events. We add their syntactic functions to their best matching roles. This expands the coverage of each learned role. Varying the 200 amount does not lead to wide variation in extraction performance. Once induced, the roles are evaluated by their entity extraction performance in Section 5.

4.4 Template Evaluation

We now compare our learned templates to those hand-created by human annotators for the MUC-4 terrorism corpus. The corpus contains 6 template types, but two of them occur in only 4 and 14 of the 1300 training documents. We thus only evaluate the 4 main templates (*bombing*, *kidnapping*, *attack*, and *arson*). The gold slots are shown in figure 3.

[3]Physical objects are defined as non-person physical objects

	Bombing	**Kidnap**	**Attack**	**Arson**
Perpetrator	x	x	x	x
Victim	x	x	x	x
Target	x		x	x
Instrument	x		x	

Figure 3: Slots in the hand-crafted MUC-4 templates.

We evaluate the four learned templates that score highest in the document classification evaluation (to be described in section 5.1), aligned with their MUC-4 types. Figure 2 shows three of our four templates, and two brand new ones that our algorithm learned. Of the four templates, we learned 12 of the 13 semantic roles as created for MUC. In addition, we learned a new role not in MUC for bombings, kidnappings, and arson: the *Police* or *Authorities* role. The annotators chose not to include this in their labeling, but this knowledge is clearly relevant when understanding such events, so we consider it correct. There is one additional Bombing and one Arson role that does not align with MUC-4, marked incorrect.

We thus report 92% slot recall, and precision as 14 of 16 (88%) learned slots.

We only measure agreement with the MUC template schemas, but our system learns other events as well. We show two such examples in figure 2: the Weapons Smuggling and Election Templates.

5 Information Extraction: Slot Filling

We now present how to apply our learned templates to information extraction. This section will describe how to extract slot fillers using our templates, but without knowing which templates are correct.

We could simply use a standard IE approach, for example, creating seed words for our new learned templates. But instead, we propose a new method that obviates the need for even a limited human labeling of seed sets. We consider each learned semantic role as a potential slot, and we extract slot fillers using the syntactic functions that were previously learned. Thus, the learned syntactic patterns (e.g., the subject of *release*) serve the dual purpose of both inducing the template slots, and extracting appropriate slot fillers from text.

5.1 Document Classification

A document is labeled for a template if two different conditions are met: (1) it contains at least one trigger phrase, and (2) its average per-token conditional probability meets a strict threshold.

Both conditions require a definition of the conditional probability of a template given a token. The conditional is defined as the token's importance relative to its uniqueness across all templates. This is not the usual conditional probability definition as IR-corpora are different sizes.

$$P(t|w) = \frac{P_{IR_t}(w)}{\sum_{s \in T} P_{IR_s}(w)} \tag{8}$$

where $P_{IR_t}(w)$ is the probability of pattern w in the IR-corpus of template t.

$$P_{IR_t}(w) = \frac{C_t(w)}{\sum_v C_t(v)} \tag{9}$$

where $C_t(w)$ is the number of times word w appears in the IR-corpus of template t. A template's trigger words are defined as words satisfying $P(t|w) > 0.2$.

	Kidnap	Bomb	Attack	Arson
Precision	.64	.83	.66	.30
Recall	.54	.63	.35	1.0
F1	**.58**	**.72**	**.46**	**.46**

Figure 4: Document classification results on test.

Trigger phrases are thus template-specific patterns that are highly indicative of that template.

After identifying triggers, we use the above definition to score a document with a template. A document is labeled with a template if it contains at least one trigger, and its average word probability is greater than a parameter optimized on the training set. A document can be (and often is) labeled with multiple templates.

Finally, we label the sentences that contain triggers and use them for extraction in section 5.2.

5.1.1 Experiment: Document Classification

The MUC-4 corpus links templates to documents, allowing us to evaluate our document labels. We treat each link as a gold label (kidnap, bomb, or attack) for that document, and documents can have multiple labels. Our learned clusters naturally do not have MUC labels, so we report results on the four clusters that score highest with each label.

Figure 4 shows the document classification scores. The bombing template performs best with an F1 score of .72. Arson occurs very few times, and Attack is lower because it is essentially an agglomeration of diverse events (discussed later).

5.2 Entity Extraction

Once documents are labeled with templates, we next extract entities into the template slots. Extraction occurs in the trigger sentences from the previous section. The extraction process is two-fold:

1. Extract all NPs that are arguments of patterns in the template's induced roles.
2. Extract NPs whose heads are observed frequently with one of the roles (e.g., 'bomb' is seen with Instrument relations in figure 2).

Take the following MUC-4 sentence as an example:

> *The two bombs were planted with the exclusive purpose of intimidating the owners of...*

The verb *plant* is in our learned bombing cluster, so step (1) will extract its passive subject *bombs* and map it to the correct instrument role (see figure 2). The human target, *owners*, is missed because *intimidate* was not learned. However, if *owner* is in the selectional preferences of the learned 'human target' role, step (2) correctly extracts it into that role.

These are two different, but complementary, views of semantic roles. The first is that a role is defined by the set of syntactic relations that describe it. Thus, we find all role relations and save their arguments (pattern extraction). The second view is that a role is defined by the arguments that fill it. Thus, we extract all arguments that filled a role in training, regardless of their current syntactic environment.

Finally, we filter extractions whose WordNet or named entity label does not match the learned slot's type (e.g., a Location does not match a Person).

6 Standard Evaluation

We trained on the 1300 documents in the MUC-4 corpus and tested on the 200 document TST3 and TST4 test set. We evaluate the four string-based slots: perpetrator, physical target, human target, and instrument. We merge MUC's two perpetrator slots (individuals and orgs) into one gold Perpetrator slot. As in Patwardhan and Riloff (2007; 2009), we ignore missed optional slots in computing recall. We induced clusters in training, performed IR, and induced the slots. We then extracted entities from the test documents as described in section 5.2.

The standard evaluation for this corpus is to report the F1 score for slot type accuracy, ignoring the template type. For instance, a perpetrator of a bombing and a perpetrator of an attack are treated the same. This allows supervised classifiers to train on all perpetrators at once, rather than template-specific learners. Although not ideal for our learning goals, we report it for comparison against previous work.

Several supervised approaches have presented results on MUC-4, but unfortunately we cannot compare against them. Maslennikov and Chua (2006; 2007) evaluated a random subset of test (they report .60 and .63 F1), and Xiao et al. (2004) did not evaluate all slot types (they report .57 F1).

Figure 5 thus shows our results with previous work that is comparable: the fully supervised and weakly supervised approaches of Patwardhan and Riloff (2009; 2007). We give two numbers for our system: mapping one learned template to Attack, and mapping five. Our learned templates for Attack have a different granularity than MUC-4. Rather than one broad Attack type, we learn several: Shooting, Murder, Coup, General Injury, and Pipeline Attack. We see these subtypes as strengths of our algorithm, but it misses the MUC-4 granularity of Attack. We thus show results when we apply the best five learned templates to Attack, rather than just one. The final F1 with these Attack subtypes is .40.

Our precision is as good as (and our F1 score near) two algorithms that require knowledge of the templates and/or labeled data. Our algorithm instead learned this knowledge without such supervision.

	P	R	F1
Patwardhan & Riloff-09 : Supervised	48	59	53
Patwardhan & Riloff-07 : Weak-Sup	42	48	44
Our Results (1 attack)	48	25	33
Our Results (5 attack)	**44**	**36**	**40**

Figure 5: MUC-4 extraction, ignoring template type.

F1 Score	Kidnap	Bomb	Arson	Attack
Results	.53	.43	.42	.16 / .25

Figure 6: Performance of individual templates. Attack compares our 1 vs 5 best templates.

7 Specific Evaluation

In order to more precisely evaluate each learned template, we also evaluated per-template performance. Instead of merging all slots across all template types, we score the slots within each template type. This is a stricter evaluation than Section 6; for example, bombing victims assigned to attacks were previously deemed correct[4].

Figure 6 gives our results. Three of the four templates score at or above .42 F1, showing that our lower score from the previous section is mainly due to the Attack template. Arson also unexpectedly

[4]We do not address the task of template instance identification (e.g., splitting two bombings into separate instances). This requires deeper discourse analysis not addressed by this paper.

	Precision	Recall	F1
Kidnap	.82	.47	.60 (+.07)
Bomb	.60	.36	.45 (+.02)
Arson	1.0	.29	.44 (+.02)
Attack	.36	.09	.15 (0.0)

Figure 7: Performance of each template type, but only evaluated on documents labeled with each type. All others are removed from test. The parentheses indicate F1 gain over evaluating on all test documents (figure 6).

scored well. It only occurs in 40 documents overall, suggesting our algorithm works with little evidence.

Per-template performace is good, and our .40 overall score from the previous section illustrates that we perform quite well in comparison to the .44-.53 range of weakly and fully supervised results.

These evaluations use the standard TST3 and TST4 test sets, including the documents that are not labeled with any templates. 74 of the 200 test documents are unlabeled. In order to determine where the system's false positives originate, we also measure performance only on the 126 test documents that have at least one template. Figure 7 presents the results on this subset. Kidnap improves most significantly in F1 score (7 F1 points absolute), but the others only change slightly. Most of the false positives in the system thus do not originate from the unlabeled documents (the 74 unlabeled), but rather from extracting incorrect entities from correctly identified documents (the 126 labeled).

8 Discussion

Template-based IE systems typically assume knowledge of the domain and its templates. We began by showing that domain knowledge isn't necessarily required; we learned the MUC-4 template structure with surprising accuracy, learning new semantic roles and several new template structures. We are the first to our knowledge to automatically induce MUC-4 templates. It is possible to take these learned slots and use a previous approach to IE (such as seed-based bootstrapping), but we presented an algorithm that instead uses our learned syntactic patterns. We achieved results with comparable precision, and an F1 score of .40 that approaches prior algorithms that rely on hand-crafted knowledge.

The extraction results are encouraging, but the template induction itself is a central contribution of this work. Knowledge induction plays an important role in moving to new domains and assisting users who may not know what a corpus contains. Recent work in Open IE learns atomic relations (Banko et al., 2007b), but little work focuses on structured scenarios. We learned more templates than just the main MUC-4 templates. A user who seeks to know what information is in a body of text would instantly recognize these as key templates, and could then extract the central entities.

We hope to address in the future how the algorithm's unsupervised nature hurts recall. Without labeled or seed examples, it does not learn as many patterns or robust classifiers as supervised approaches. We will investigate new text sources and algorithms to try and capture more knowledge. The final experiment in figure 7 shows that perhaps new work should first focus on pattern learning and entity extraction, rather than document identification.

Finally, while our pipelined approach (template induction with an IR stage followed by entity extraction) has the advantages of flexibility in development and efficiency, it does involve a number of parameters. We believe the IR parameters are quite robust, and did not heavily focus on improving this stage, but the two clustering steps during template induction require parameters to control stopping conditions and word filtering. While all learning algorithms require parameters, we think it is important for future work to focus on removing some of these to help the algorithm be even more robust to new domains and genres.

Acknowledgments

This work was supported by the National Science Foundation IIS-0811974, and this material is also based upon work supported by the Air Force Research Laboratory (AFRL) under prime contract no. FA8750-09-C-0181. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of the Air Force Research Laboratory (AFRL). Thanks to the Stanford NLP Group and reviewers for helpful suggestions.

References

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet project. In Christian Boitet and Pete Whitelock, editors, *ACL-98*, pages 86–90, San Francisco, California. Morgan Kaufmann Publishers.

Michele Banko, Michael J Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. 2007a. Learning relations from the web. In *Proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI)*.

Michele Banko, Michael J Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. 2007b. Open information extraction from the web. In *Proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI)*.

David Blei, Andrew Ng, and Michael Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*.

Razvan Bunescu and Raymond Mooney. 2004. Collective information extraction with relational markov networks. In *Proceedings of the Association of Computational Linguistics (ACL)*, pages 438–445.

Andrew Carlson, J. Betteridge, R.C. Wang, E.R. Hruschka Jr., and T.M. Mitchell. 2010. Coupled semi-supervised learning for information extraction. In *Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM)*.

Nathanael Chambers and Dan Jurafsky. 2008. Unsupervised learning of narrative event chains. In *Proceedings of the Association of Computational Linguistics (ACL)*, Hawaii, USA.

Nathanael Chambers and Dan Jurafsky. 2009. Unsupervised learning of narrative schemas and their participants. In *Proceedings of the Association of Computational Linguistics (ACL)*, Columbus, Ohio.

Hai Leong Chieu, Hwee Tou Ng, and Yoong Keok Lee. 2003. Closing the gap: Learning-based information extraction rivaling knowledge-engineering methods. In *Proceedings of the Association of Computational Linguistics (ACL)*.

Nancy Chinchor, David Lewis, and Lynette Hirschman. 1993. Evaluating message understanding systems: an analysis of the third message understanding conference. *Computational Linguistics*, 19:3:409–449.

Katrin Erk and Sebastian Pado. 2008. A structured vector space model for word meaning in context. In *Proceedings of the 2008 Conference on Empirical Methods on Natural Language Processing (EMNLP)*.

Elena Filatova, Vasileios Hatzivassiloglou, and Kathleen McKeown. 2006. Automatic creation of domain templates. In *Proceedings of the Association of Computational Linguistics (ACL)*.

Dayne Freitag. 1998. Toward general-purpose learning for information extraction. In *Proceedings of the Association of Computational Linguistics (ACL)*, pages 404–408.

David Graff. 2002. English gigaword. *Linguistic Data Consortium*.

Trond Grenager and Christopher D. Manning. 2006. Unsupervised discovery of a statistical verb lexicon. In *Proceedings of the the 2006 Conference on Empirical Methods on Natural Language Processing (EMNLP)*.

Shan He and Daniel Gildea. 2006. Self-training and co-training for semantic role labeling: Primary report. Technical Report 891, University of Rochester.

Heng Ji and Ralph Grishman. 2008. Refining event extraction through unsupervised cross-document inference. In *Proceedings of the Association of Computational Linguistics (ACL)*.

Niels Kasch and Tim Oates. 2010. Mining script-like structures from the web. In *Proceedings of NAACL HLT*, pages 34–42.

Joel Lang and Mirella Lapata. 2010. Unsupervised induction of semantic roles. In *Proceedings of the North American Association of Computational Linguistics*.

Mstislav Maslennikov and Tat-Seng Chua. 2007. Automatic acquisition of domain knowledge for information extraction. In *Proceedings of the Association of Computational Linguistics (ACL)*.

Siddharth Patwardhan and Ellen Riloff. 2007. Effective ie with semantic affinity patterns and relevant regions. In *Proceedings of the 2007 Conference on Empirical Methods on Natural Language Processing (EMNLP)*.

Siddharth Patwardhan and Ellen Riloff. 2009. A unified model of phrasal and sentential evidence for information extraction. In *Proceedings of the 2009 Conference on Empirical Methods on Natural Language Processing (EMNLP)*.

Lisa Rau, George Krupka, Paul Jacobs, Ira Sider, and Lois Childs. 1992. Ge nltoolset: Muc-4 test results and analysis. In *Proceedings of the Message Understanding Conference (MUC-4)*, pages 94–99.

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

Ellen Riloff, Janyce Wiebe, and William Phillips. 2005. Exploiting subjectivity classification to improve information extraction. In *Proceedings of AAAI-05*.

Roger C. Schank and Robert P. Abelson. 1977. *Scripts, plans, goals and understanding*. Lawrence Erlbaum.

Yusuke Shinyama and Satoshi Sekine. 2006. Preemptive ie using unrestricted relation discovery. In *Proceedings of NAACL*.

Kiyoshi Sudo, Satoshi Sekine, and Ralph Grishman. 2003. An improved extraction pattern representation model for automatic ie pattern acquisition. In *Proceedings of the Association of Computational Linguistics (ACL)*, pages 224–231.

Beth M. Sundheim. 1991. Third message understanding evaluation and conference (muc-3): Phase 1 status report. In *Proceedings of the Message Understanding Conference*.

Mihai Surdeanu, Jordi Turmo, and Alicia Ageno. 2006. A hybrid approach for the acquisition of information extraction patterns. In *Proceedings of the EACL Workshop on Adaptive Text Extraction and Mining*.

Robert S. Swier and Suzanne Stevenson. 2004. Unsupervised semantic role labelling. In *Proceedings of the 2004 Conference on Empirical Methods on Natural Language Processing (EMNLP)*.

Jing Xiao, Tat-Seng Chua, and Hang Cui. 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)*.

Roman Yangarber, Ralph Grishman, Pasi Tapanainen, and Silja Huttunen. 2000. Automatic acquisition of domain knowledge for information extraction. In *COLING*, pages 940–946.