Probabilistic Frame Induction

Jackie Chi Kit Cheung*
Department of Computer Science
University of Toronto
Toronto, ON, M5S 3G4, Canada
jcheung@cs.toronto.edu

Hoifung Poon
One Microsoft Way
Microsoft Research
Redmond, WA 98052, USA
hoifung@microsoft.com

Lucy Vanderwende
One Microsoft Way
Microsoft Research
Redmond, WA 98052, USA
lucyv@microsoft.com

Abstract

In natural-language discourse, related events tend to appear near each other to describe a larger scenario. Such structures can be formalized by the notion of a *frame* (a.k.a. template), which comprises a set of related events and prototypical participants and event transitions. Identifying frames is a prerequisite for information extraction and natural language generation, and is usually done manually. Methods for inducing frames have been proposed recently, but they typically use ad hoc procedures and are difficult to diagnose or extend. In this paper, we propose the first probabilistic approach to frame induction, which incorporates frames, events, and participants as latent topics and learns those frame and event transitions that best explain the text. The number of frame components is inferred by a novel application of a split-merge method from syntactic parsing. In end-to-end evaluations from text to induced frames and extracted facts, our method produces state-of-the-art results while substantially reducing engineering effort.

1 Introduction

Events with causal or temporal relations tend to occur near each other in text. For example, a BOMBING scenario in an article on terrorism might begin with a DETONATION event, in which terrorists set off a bomb. Then, a DAMAGE event might ensue to describe the resulting destruction and any casualties, followed by an INVESTIGATION event covering subsequent police investigations. Afterwards, the BOMBING scenario may transition into a CRIMINAL-PROCESSING scenario, which begins with police catching the terrorists, and proceeds to a trial, sentencing, etc. A common set of participants serves as the event arguments; e.g., the agent (or subject) of DETONATION is often the same as the theme (or object) of INVESTIGATION and corresponds to a PERPETRATOR.

Such structures can be formally captured by the notion of a *frame* (a.k.a. template, scenario), which consists of a set of *events* with prototypical transitions, as well as a set of *slots* representing the common participants. Identifying frames is an explicit or implicit prerequisite for many NLP tasks. Information extraction, for example, stipulates the types of events and slots that are extracted for a frame or template. Online applications such as dialogue systems and personal-assistant applications also model users' goals and subgoals using frame-like representations. In natural-language generation, frames are often used to represent contents to be expressed as well as to support surface realization.

Until recently, frames and related representations have been manually constructed, which has limited their applicability to a relatively small number of domains and a few slots within a domain. Furthermore, additional manual effort is needed after the frames are defined in order to extract frame components from text (e.g., in annotating examples and designing features to train a supervised learning model). This paradigm makes generalizing across tasks difficult, and might suffer from annotator bias.

Recently, there has been increasing interest in au-

*This research was undertaken during the author's internship at Microsoft Research.

tomatically inducing frames from text. A notable example is Chambers and Jurafsky (2011), which first clusters related verbs to form frames, and then clusters the verbs' syntactic arguments to identify slots. While Chambers and Jurafsky (2011) represents a major step forward in frame induction, it is also limited in several aspects. The clustering used ad hoc steps and customized similarity metrics, as well as an additional retrieval step from a large external text corpus for slot generation. This makes it hard to replicate their approach or adapt it to new domains. Lacking a coherent model, it is also difficult to incorporate additional linguistic insights and prior knowledge.

In this paper, we present PROFINDER (PRObabilistic Frame INDucER), the first probabilistic approach to frame induction. PROFINDER defines a joint distribution over the words in a document and their frame assignments by modeling frame and event transitions, correlations among events and slots, and their surface realizations. Given a set of documents, PROFINDER outputs a set of induced frames with learned parameters, as well as the most probable frame assignments that can be used for event and entity extraction. The numbers of events and slots are dynamically determined by a novel application of the split-merge approach from syntactic parsing (Petrov et al., 2006). In end-to-end evaluations from text to entity extraction using standard MUC and TAC datasets, PROFINDER achieved state-of-the-art results while significantly reducing engineering effort and requiring no external data.

2 Related Work

In information extraction and other semantic processing tasks, the dominant paradigm requires two stages of manual effort. First, the target representation is defined manually by domain experts. Then, manual effort is required to construct an extractor or to annotate examples to train a machine-learning system. Recently, there has been a burgeoning body of work in reducing such manual effort. For example, a popular approach to reduce annotation effort is bootstrapping from seed examples (Patwardhan and Riloff, 2007; Huang and Riloff, 2012). However, this still requires prespecified frames or templates, and selecting seed words is often a challenging task (Curran et al., 2007). Filatova et al. (2006) construct simple domain templates by mining verbs and the named entity type of verbal arguments that are topical, whereas Shinyama and Sekine (2006) identify query-focused slots by clustering common named entities and their syntactic contexts. Open IE (Banko and Etzioni, 2008) limits the manual effort to designing a few domain-independent relation patterns, which can then be applied to extract relational triples from text. While extremely scalable, this approach can only extract atomic factoids within a sentence, and the resulting triples are noisy, non-canonicalized text fragments.

More relevant to our approach is the recent work in unsupervised semantic induction, such as unsupervised semantic parsing (Poon and Domingos, 2009), unsupervised semantical role labeling (Swier and Stevenson, 2004) and induction (Lang and Lapata, 2011, e.g.), and slot induction from web search logs (Cheung and Li, 2012). As in PROFINDER, they model distributional contexts for slots and roles. However, these approaches focus on the semantics of independent sentences or queries, and do not capture discourse-level dependencies.

The modeling of frame and event transitions in PROFINDER is similar to a sequential topic model (Gruber et al., 2007), and is inspired by the successful applications of such topic models in summarization (Barzilay and Lee, 2004; Daumé III and Marcu, 2006; Haghighi and Vanderwende, 2009, inter alia). There are, however, two main differences. First, PROFINDER contains not a single sequential topic model, but two (for frames and events, respectively). In addition, it also models the interdependencies among events, slots, and surface text, which is analogous to the USP model (Poon and Domingos, 2009). PROFINDER can thus be viewed as a novel combination of state-of-the-art models in unsupervised semantics and discourse modeling.

In terms of aim and capability, PROFINDER is most similar to Chambers and Jurafsky (2011), which culminated from a series of work for identifying correlated events and arguments in narratives (Chambers and Jurafsky, 2008; Chambers and Jurafsky, 2009). By adopting a probabilistic approach, PROFINDER has a sound theoretical underpinning, and is easy to modify or extend. For example, in Section 3, we show how PROFINDER can easily be

augmented with additional linguistically-motivated features. Likewise, PROFINDER can easily be used as a semi-supervised system if some slot designations and labeled examples are available.

The idea of representing and capturing stereotypical knowledge has a long history in artificial intelligence and psychology, and has assumed various names such as *frames* (Minsky, 1974), *schemata* (Rumelhart, 1975), and *scripts* (Schank and Abelson, 1977). In the linguistics and computational linguistics communities, frame semantics (Fillmore, 1982) uses frames as the central representation of word meaning, culminating in the development of FrameNet (Baker et al., 1998), which contains over 1000 manually annotated frames. A similarly rich lexical resource is the MindNet project (Richardson et al., 1998). Our notion of frame is related to these representations, but there are also subtle differences. For example, Minsky's frame emphasizes *inheritance*, which we do not model in this paper[1]. As in semantic role labeling, FrameNet focuses on semantic roles and does not model event or frame transitions, so the scope of its frames is often no more than an event in our model. Perhaps the most similar to our frame is Roger Schank's scripts, which capture prototypical events and participants in a scenario such as restaurant dining. In their approach, however, scripts are manually defined, making it hard to generalize. In this regard, our work may be viewed as an attempt to revive a long tradition in AI and linguistics, by leveraging the recent advances in computational power, NLP, and machine learning.

3 Probabilistic Frame Induction

In this section, we present PROFINDER, a probabilistic model for frame induction. Let $\mathcal{F}$ be a set of frames, where each frame $F = (E_F, S_F)$ comprises a unique set of events E_F and slots S_F. Given a document D and a word w in D, $Z_w = (f, e)$ represents an assignment of w to frame $f \in \mathcal{F}$ and frame element $e \in E_f \cup S_f$. At the heart of PROFINDER is a generative model $P_\theta(D, Z)$ that defines a joint distribution over document D and the frame assignment to its words Z. Given a set of documents $\mathcal{D}$, frame induction in PROFINDER amounts to determining the number of events and slots in each frame, as well as learning the parameters θ by summing out the latent assignments Z to maximize the likelihood of the document set

$$\prod_{D \in \mathcal{D}} P_\theta(D).$$

The induced frames identify the key event structures in the document set. Additionally, PROFINDER can conduct event and entity extraction by computing the most probable frame assignment Z. In the remainder of the section, we first present the base model for PROFINDER. We then introduce several linguistically motivated refinements, as well as efficient algorithms for learning and inference in PROFINDER.

3.1 Base Model

The probabilistic formulation of PROFINDER makes it extremely flexible for incorporating linguistic intuition and prior knowledge. In this paper, we design our PROFINDER model to capture three types of dependencies.

Frame transitions between clauses A sentence contains one or more clauses, each of which is a minimal unit expressing a proposition. A clause is unlikely to straddle different frames, so we stipulate that the words in a clause be assigned to the same frame. On the other hand, frame transitions can happen between clauses, and we adopt the common Markov assumption that the frame of a clause only depends on the previous clause in the document. Clauses are automatically extracted from the dependency parse and further decomposed into an *event head* and its syntactic arguments.

Event transitions within a frame Events tend to transition into related events in the same frame, as determined by their causal or temporal relations. Each clause is assigned an event compatible with its frame assignment (i.e., the event is in the given frame). Like frame transitions, we assume that the event assignment of a clause depends only on the event of the previous clause.

Emission of event heads and slot words Similar to topics in topic models, each event determines

[1]This should be a straightforward extension — using the split-and-merge approach, PROFINDER already produces a hierarchy of events and slots in learning, although currently it makes no use of the intermediate levels.

a multinomial from which the event head is generated; e.g., a DETONATION event might use verbs such as *detonate, set off* or nouns such as *denotation, bombing* as its event head. Additionally, as in USP (Poon and Domingos, 2009), an event also contains a multinomial of slots for each of its argument types[2]; e.g., the agent argument of a DETONATION event is generally the PERPETRATOR slot of the BOMBING frame. Finally, each slot has its own multinomials for generating the argument head and dependency label, regardless of the event.

Formally, let D be a document and $C_1, \cdots, C_l$ be its clauses, the PROFINDER model is defined by

$$\begin{aligned} P_\theta(D, Z) = & P_{\texttt{F-INIT}}(F_1) \times \prod_i P_{\texttt{F-TRAN}}(F_{i+1}|F_i) \\ & \times P_{\texttt{E-INIT}}(E_1|F_1) \\ & \times \prod_i P_{\texttt{E-TRAN}}(E_{i+1}|E_i, F_{i+1}, F_i) \\ & \times \prod_i P_{\texttt{E-HEAD}}(e_i|E_i) \\ & \times \prod_{i,j} P_{\texttt{SLOT}}(S_{i,j}|E_{i,j}, A_{i,j}) \\ & \times \prod_{i,j} P_{\texttt{A-HEAD}}(a_{i,j}|S_{i,j}) \\ & \times \prod_{i,j} P_{\texttt{A-DEP}}(dep_{i,j}|S_{i,j}) \end{aligned}$$

Here, F_i, E_i denote the frame and event assignment to clause C_i, respectively, and e_i denotes the event head. For the j-th argument of clause i, $S_{i,j}$ denotes the slot assignment, $A_{i,j}$ the argument type, $a_{i,j}$ the head word, and $dep_{i,j}$ the dependency from the event head. $P_{\texttt{E-TRAN}}(E_{i+1}|E_i, F_{i+1}, F_i) = P_{\texttt{E-INIT}}(E_{i+1}|F_{i+1})$ if $F_{i+1} \neq F_i$.

Essentially, PROFINDER combines a frame HMM with an event HMM, where the first models frame transition and emits events, and the second models event transition within a frame and emits argument slots.

3.2 Model refinements

The base model captures the main dependencies in event narrative, but it can be easily extended to leverage additional linguistic intuition. PROFINDER incorporates three such refinements.

Background frame Event narratives often contain interjections of general content common to all frames. For example, in newswire articles, ATTRIBUTION is commonplace to describe who said or reported a particular quote or fact. To avoid contaminating frames with generic content, we introduce a background frame with its own events, slots, and emission distributions, and a binary switch variable $B_i \in \{BKG, CNT\}$ that determines whether clause i is generated from the actual content frame F_i (CNT) or background (BKG). We also stipulate that if BKG is chosen, the nominal frame stays the same as the previous clause.

Stickiness in frame and event transitions Prior work has demonstrated that promoting topic coherence in natural-language discourse helps discourse modeling (Barzilay and Lee, 2004). We extend PROFINDER to leverage this intuition by incorporating a "stickiness" prior (Haghighi and Vanderwende, 2009) to encourage neighboring clauses to stay in the same frame. Specifically, along with introducing the background frame, the frame transition component now becomes

$$P_{\texttt{F-TRAN}}(F_{i+1}|F_i, B_{i+1}) = \quad (1)$$
$$\begin{cases} \mathbf{1}(F_{i+1} = F_i), & \text{if } B_{i+1} = BKG \\ \beta \mathbf{1}(F_{i+1} = F_i) + & \\ (1-\beta) P_{\texttt{F-TRAN}}(F_{i+1}|F_i), & \text{if } B_{i+1} = CNT \end{cases}$$

where β is the stickiness parameter, and the event transition component correspondingly becomes

$$P_{\texttt{E-TRAN}}(E_{i+1}|E_i, F_{i+1}, F_i, B_{i+1}) = \quad (2)$$
$$\begin{cases} \mathbf{1}(E_{i+1} = E_i), & \text{if } B_{i+1} = BKG \\ P_{\texttt{E-TRAN}}(E_{i+1}|E_i), & \text{if } B_{i+1} = CNT, F_i = F_{i+1} \\ P_{\texttt{E-INIT}}(E_{i+1}), & \text{if } B_{i+1} = CNT, F_i \neq F_{i+1} \end{cases}$$

Argument dependencies as caseframes As noticed in previous work such as Chambers and Jurafsky (2011), the combination of an event head and a dependency relation often gives a strong signal of the slot that is indicated. For example, $bomb > nsubj$ (subject argument of *bomb*) often indicates a PERPETRATOR. Thus, rather than simply emitting

[2]USP generates the argument types along with events from clustering. For simplicity, in PROFINDER we simply classify a syntactic argument into subject, object, and prepositional object, according to its Stanford dependency to the event head.

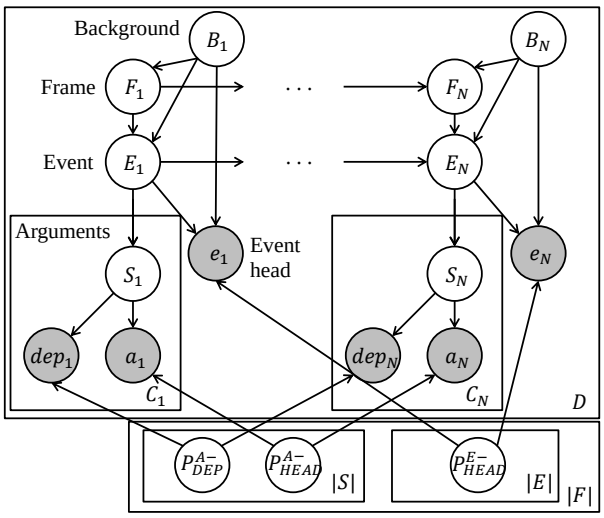

Figure 1: Graphical representation of our model. Hyperparameters, the stickiness factor, and the frame and event initial and transition distributions are not shown for clarity.

the dependency from the event head to an event argument $dep_{i,j}$, our model instead emits the pair of event head and dependency relation, which we call a *caseframe* following Bean and Riloff (2004).

3.3 Full generative story

To summarize, the distributions that are learned by our model are the default distributions $P_{\texttt{BKG}}(B)$, $P_{\texttt{F-INIT}}(F)$, $P_{\texttt{E-INIT}}(E)$; the transition distributions $P_{\texttt{F-TRAN}}(F_{i+1}|F_i)$, $P_{\texttt{E-TRAN}}(E_{i+1}|E_i)$; and the emission distributions $P_{\texttt{SLOT}}(S|E,A,B)$, $P_{\texttt{E-HEAD}}(e|E,B)$, $P_{\texttt{A-HEAD}}(a|S)$, $P_{\texttt{A-DEP}}(dep|S)$. We used additive smoothing with uniform Dirichlet priors for all the multinomials. The overall generative story of our model is as follows:

1. Draw a Bernoulli distribution for $P_{\texttt{BKG}}(B)$
2. Draw the frame, event, and slot distributions
3. Draw an event head emission distribution $P_{\texttt{E-HEAD}}(e|E,B)$ for each frame including the background frame
4. Draw event argument lemma and caseframe emission distributions for each slot in each frame including the background frame
5. For each clause in each document, generate the clause-internal structure.

The clause-internal structure at clause i is generated by the following steps:

1. Generate whether this clause is background ($B_i \in \{CNT, BKG\} \sim P_{\texttt{BKG}}(B)$)
2. Generate the frame F_i and event E_i from $P_{\texttt{F-INIT}}(F)$, $P_{\texttt{E-INIT}}(E)$, or according to equations 1 and 2
3. Generate the observed event head e_i from $P_{\texttt{E-HEAD}}(e_i|E_i)$.
4. For each event argument:
   (a) Generate the slot $S_{i,j}$ from $P_{\texttt{SLOT}}(S|E,A,B)$.
   (b) Generate the dependency/caseframe emission $dep_{i,j} \sim P_{\texttt{A-DEP}}(dep|S)$ and the lemma of the head word of the event argument $a_{i,j} \sim P_{\texttt{A-HEAD}}(a|S)$.

3.4 Learning and Inference

Our generative model admits efficient inference by dynamic programming. In particular, after collapsing the latent assignment of frame, event, and background into a single hidden variable for each clause, the expectation and most probable assignment can be computed using standard forward-backward and Viterbi algorithms on fixed tree structures.

Parameter learning can be done using EM by alternating the computation of expected counts and the maximization of multinomial parameters. In particular, PROFINDER uses incremental EM, which has been shown to have better and faster convergence properties than standard EM (Liang and Klein, 2009).

Determining the optimal number of events and slots is challenging. One solution is to adopt a nonparametric Bayesian method by incorporating a hierarchical prior over the parameters (e.g., a Dirichlet process). However, this approach can impose unrealistic restrictions on the model choice and result in intractability which requires sampling or approximate inference to overcome. Additionally, EM learning can suffer from local optima due to its nonconvex learning objective, especially when dealing with a large number hidden states without a good initialization.

To address these issues, we adopt a novel application of the split-merge method previously used in syntactic parsing for inferring refined latent syntactic categories (Petrov et al., 2006). First, the model is initialized with a number of frames, which is a hyperparameter, and each frame is associated with

one event and two slots. Starting from this minimal structure, EM training begins. After a number of iterations, each event and slot state is "split" in two; that is, each original state now becomes two new states. Each of the new states is generated with half of the probability of the original, and contains a duplicate of the associated emission distributions. Some perturbation is then added to the probabilities to break symmetry. After splitting, we merge back a portion of the newly split events and slots that result in the least improvement in the likelihood of the training data. For more details on split-merge, see Petrov et al. (2006)

By adjusting the number of split-merge cycles and the merge parameters, our model learns the number of events and slots in a dynamical fashion that is tailored to the data. Moreover, our model starts with a small number of frame elements, which reduces the number of local optima and facilitates initial learning. After each split, the subsequent learning starts with (a perturbed version of) the previously learned parameters, which makes a good initialization that is crucial for EM. Finally, it is also compatible with the hierarchical nature of events and slots. For example, slots can first be coarsely split into persons versus locations, and later refined into subcategories such as perpetrators and victims.

4 MUC-4 Entity Extraction Experiments

We first evaluate our model on a standard entity extraction task, using the evaluation settings from Chambers and Jurafsky (2011) (henceforth, C&J) to enable a head-to-head comparison. Specifically, we use the MUC-4 data set (1992) , which contains 1300 training and development documents on terrorism in South America, with 200 additional documents for testing. MUC-4 contains four templates: ATTACK, KIDNAPPING, BOMBING, and ARSON.[3] All templates share the same set of predefined slots, with the evaluation focusing on the following four: PERPETRATOR, PHYSICAL TARGET, HUMAN TARGET, and INSTRUMENT.

For each slot in a MUC template, the system first identifies an induced slot that best maps to it by F_1 on the development set. As in C&J, template is ignored in final evaluation, so all the clusters that belong to the same slot are then merged across the templates; e.g., the PERPETRATOR clusters for KIDNAPPING and BOMBING are merged. The final precision, recall, and F_1 are computed based on these merged clusters. Correctness is determined by matching head words, and slots marked as optional in MUC are ignored when computing recall. All hyperparameters are tuned on the development set (see Appendix A for their values).

Named entity type Named entity type is a useful feature to filter out entities for particular slots; e.g. a location cannot be an INSTRUMENT. We thus divide each induced cluster into four clusters by named entity type before performing the mapping, following C&J's heuristic and using a named entity recognizer and word lists derived from WordNet: PERSON/ORGANIZATION, PHYSICAL OBJECT, LOCATION, and OTHER.

Document classification The MUC-4 dataset contains many documents that have words related to MUC slots (e.g., *plane* and *aviation*), but are not about terrorism. To reduce precision errors, C&J first filtered irrelevant documents based on the specificity of event heads to learned frames. To estimate the specificity, they used additional data retrieved from a large external corpus. In PROFINDER, however, specificity can be easily estimated using the probability distributions learned during training. In particular, we define the probability of an event head in a frame j as:

$$P_F(w) = \sum_{E_F \in F} P_{\texttt{E-HEAD}}(w|E)/|F|, \qquad (3)$$

and the probability of a frame given an event head as:

$$P(F|w) = P_F(w)/\sum_{F' \in \mathcal{F}} P_{F'}(w). \qquad (4)$$

We then follow the rest of C&J's procedure to score each learned frame with each MUC document. Specifically, a document is mapped to a frame if the average $P_F(w)$ in the document is above a threshold and the document contains at least one *trigger word* w' with $P(F|w') > 0.2$. The threshold and the induced frame were determined on the development set, and were used to filter irrelevant documents in the test set.

[3] Two other templates have negligible counts and are ignored as in C&J.

Unsupervised methods	P	R	F_1
PROFINDER (This work)	32	**37**	**34**
Chambers and Jurafsky (2011)	**48**	25	33
With additional information			
PROFINDER +doc. classification	41	44	43
C&J 2011 +granularity	44	36	40

Table 1: Results on MUC-4 entity extraction. C&J 2011 +granularity refers to their experiment in which they mapped one of their templates to five learned clusters rather than one.

Results Compared to C&J, PROFINDER is conceptually much simpler, using a single probabilistic model and standard learning and inference algorithms, and not requiring multiple processing steps or customized similarity metrics. It only used the data in MUC-4, whereas C&J required additional text to be retrieved from a large external corpus (Gigaword (Graff et al., 2005)) for each event cluster. It currently does not make use of coreference information, whereas C&J did. Remarkably, despite all these, PROFINDER was still able to outperform C&J on entity extraction, as shown in Table 1. We also evaluated PROFINDER's performance assuming perfect document classification (+doc. classification). This led to a substantially higher precision, suggesting that further improvement is possible from better document classification.

Figure 2 shows part of a frame learned by PROFINDER, which includes some slots and events annotated in MUC. PROFINDER is also able to identify events and slots not annotated in MUC, a desirable characteristic of unsupervised methods. For example, it found a DISCUSSION event, an ARREST event (*call, arrest, express, meet, charge*), a PEACE AGREEMENT slot (*agreement, rights, law, proposal*), and an AUTHORITIES slot (*police, government, force, command*). The background frame was able to capture many verbs related to attribution, such as *say, continue, add, believe*, although it missed *report*.

Event: Attack	**Event: Discussion**
report, participate, kidnap, kill, release	*hold, meeting, talk, discuss, investigate*
Slot: Perpetrator	**Slot: Victim**
PERSON/ORG	PERSON/ORG
Words: *guerrilla, police, source, person, group*	**Words:** *people, priest, leader, member, judge*
Caseframes:	**Caseframes:**
report>nsubj,	*kill>dobj,*
kidnap>nsubj,	*murder>dobj,*
kill>nsubj,	*release>dobj,*
participate>nsubj,	*report>dobj,*
release>nsubj	*kidnap>dobj*

Figure 2: A partial frame learned by PROFINDER from the MUC-4 data set, with the most probable emissions for each event and slot. Labels are assigned by the authors for readability.

5 Evaluating Frame Induction Using Guided Summarization Templates

The MUC-4 dataset was originally designed for information extraction and focuses on a limited number of template and slot types. To evaluate PROFINDER's capabilities in generalizing to a greater variety of text, we designed and conducted a novel evaluation based on the TAC guided-summarization dataset. This evaluation was inspired by the connection between summarization and information extraction (White et al., 2001), and reflects a conceptualization of summarization as inducing and extracting structured information from source text. Essentially, we adapted the TAC summarization annotation to create gold-standard slots, and used them to evaluate entity extraction as in MUC-4.

Dataset We used the TAC 2010 guided-summarization dataset in our experiments (Owczarzak and Dang, 2010). This data set consists of text from five domains (termed *categories* in TAC), each with a template defined by TAC organizers. In total, there are 46 document clusters (termed *topics* in TAC), each of which contains 20 documents and has eight human-written summaries. Each summary was manually segmented using the Pyramid method (Nenkova and Passonneau, 2004) and each segment was annotated with a slot (termed *aspect* in TAC) from the corresponding template. Figure 3 shows an example and the full set of templates is available at `http://www.nist.gov/tac/2010/Summarization/Guided-Summ.2010.guidelines.html`. In

(a) **Accidents and Natural Disasters:**
WHAT: what happened
WHEN: date, time, other temporal markers
WHERE: physical location
WHY: reasons for accident/disaster
WHO_AFFECTED: casualties...
DAMAGES: ... caused by the disaster
COUNTERMEASURES: rescue efforts...

(b) *(*WHEN *During the night of July 17,) (*WHAT *a 23-foot <*WHAT *tsunami) hit the north coast of Papua New Guinea (PNG)>, (*WHY *triggered by a 7.0 undersea earthquake in the area).*

(c) WHEN: *night* WHAT: *tsunami, coast* WHY: *earthquake*

Figure 3: (a) A frame from the TAC Guided Summarization task with abbreviated slot descriptions. (b) A TAC text span, segmented into several contributors with slot labels. Note that the two WHAT contributors overlap, and are demarcated by different bracket types. (c) The entities that are extracted for evaluation.

TAC, each annotated segment (Figure 3b) is called a *contributor*.

Evaluation Method We converted the contributors into a form that is more similar to the previous MUC evaluation, so that we can fairly compare against previous work such as C&J that were designed to extract information into that form. Specifically, we extracted the head lemma from all the maximal noun phrases found in the contributor (Figure 3c) and treated them as gold-standard entity slots to extract. While this conversion may not be ideal in some cases, it simplifies the TAC slots and enables automatic evaluation. We leave the refinement of this conversion to future work, and believe it could be done by crowdsourcing.

For each TAC slot in a TAC category, we extract entities from the summaries that belong to the given TAC category. A system-induced entity is considered a match to a TAC-derived entity from the same document if the head lemma in the former matches one in the latter. Based on this matching criterion, the system-induced slots are mapped to the TAC slots in a way that achieves the best F_1 for each TAC slot. We allow a system slot to map to multiple TAC slots, due to potential overlaps in entities among TAC slots. For example, in a document about a tsunami, *earthquake* may appear both in the WHAT slot as a disaster itself, and in the CAUSE slot as a cause for the tsunami.

	1-best			*5-best*		
Systems	P	R	F_1	P	R	F_1
PROFINDER	24	**25**	**24**	21	**38**	**27**
C&J	**58**	6.1	11	**50**	12	20

Table 2: Results on TAC 2010 entity extraction with N-best mapping for $N = 1$ and $N = 5$. Intermediate values of N produce intermediate results, and are not shown for brevity.

One salient difference between TAC and MUC slots is that TAC slots are often more general than MUC slots. For example, TAC slots such as WHY and COUNTERMEASURES likely correspond to multiple slots at the granularity of MUC. As a result, we also consider mapping the N-best system-induced slots to each TAC slot, for N up to 5.

Experiments We trained PROFINDER and a reimplementation of C&J on the 920 full source texts of TAC 2010, and tested them on the 368 model summaries. We did not provide C&J's model with access to external data, in order to enable fair comparison with our model. Since all of the summary sentences are expected to be relevant, we did not conduct document or sentence relevance classification in C&J or PROFINDER. We tuned all parameters by two-fold cross validation on the summaries. We computed the overall precision, recall, and F_1 by taking a micro-average over the results for each TAC slot.

Results The results are shown in Table 2. PROFINDER substantially outperformed C&J in F_1, in both 1-best and N-best cases. As in MUC-4, the precision of C&J is higher, partly because C&J often did not do much in clustering and produced many small clusters. For example, in the 1-best setting, the average number of entities mapped to each TAC slot by C&J is 21, whereas it is 208 for PROFINDER. For both systems, the results are generally lower compared to that in MUC-4, which is expected since this task is harder given the greater diversity in frames and slots to be induced.

6 Conclusion

We have presented PROFINDER, the first probabilistic approach to frame induction and shown that it achieves state-of-the-art results on end-to-end entity extraction in standard MUC and TAC data sets. Our model is inspired by recent advances in unsupervised semantic induction and content modeling in summarization. Our probabilistic approach makes it easy to extend the model with additional linguistic insights and prior knowledge. While we have made a case for unsupervised methods and the importance of robustness across domains, our method is also amenable to semi-supervised or supervised learning if annotated data is available. In future work, we would like to further investigate frame induction evaluation, particularly in evaluating event clustering.

Acknowledgments

We would like to thank Nate Chambers for answering questions about his system. We would also like to thank Chris Quirk for help with preprocessing the MUC corpus, and the members of the NLP group at Microsoft Research for useful discussions.

Appendix A. Hyperparameter Settings

We document below the hyperparameter settings for PROFINDER that were used to generate the results in the paper.

Hyperparameter	**MUC**	**TAC**
Number of frames, $\|\mathcal{F}\|$	9	8
Frame stickiness, β	0.125	0.5
Smoothing (frames, events, slots)	0.5	2
Smoothing (emissions)	0.05	0.2
Number of split-merge cycles	4	2
Iterations per cycle	10	10

References

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet project. In *Proceedings of the 17th International Conference on Computational linguistics*.

Michele Banko and Oren Etzioni. 2008. The tradeoffs between open and traditional relation extraction. *Proceedings of ACL-08: HLT*, pages 28–36.

Regina Barzilay and Lillian Lee. 2004. Catching the drift: Probabilistic content models, with applications to generation and summarization. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004*.

David Bean and Ellen Riloff. 2004. Unsupervised learning of contextual role knowledge for coreference resolution. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004*.

Nathanael Chambers and Dan Jurafsky. 2008. Unsupervised learning of narrative event chains. In *Proceedings of ACL-08: HLT*, pages 789–797, Columbus, Ohio, June. Association for Computational Linguistics.

Nathanael Chambers and Dan Jurafsky. 2009. Unsupervised learning of narrative schemas and their participants. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. Association for Computational Linguistics.

Nathanael Chambers and Dan Jurafsky. 2011. Template-based information extraction without the templates. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 976–986, Portland, Oregon, USA, June. Association for Computational Linguistics.

Jackie C. K. Cheung and Xiao Li. 2012. Sequence clustering and labeling for unsupervised query intent discovery. In *Proceedings of the 5th ACM International Conference on Web Search and Data Mining*, pages 383–392.

James R. Curran, Tara Murphy, and Bernhard Scholz. 2007. Minimising semantic drift with mutual exclusion bootstrapping. In *Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics*.

Hal Daumé III and Daniel Marcu. 2006. Bayesian Query-Focused summarization. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 305–312, Sydney, Australia, July. Association for Computational Linguistics.

Elena Filatova, Vasileios Hatzivassiloglou, and Kathleen McKeown. 2006. Automatic creation of domain templates. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 207–214, Sydney, Australia, July. Association for Computational Linguistics.

Charles J. Fillmore. 1982. Frame semantics. *Linguistics in the Morning Calm*, pages 111–137.

David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2005. English gigaword second edition. *Linguistic Data Consortium, Philadelphia*.

Amit Gruber, Michael Rosen-Zvi, and Yair Weiss. 2007. Hidden topic markov models. *Artificial Intelligence and Statistics (AISTATS)*.

Aria Haghighi and Lucy Vanderwende. 2009. Exploring content models for multi-document summarization. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 362–370, Boulder, Colorado, June. Association for Computational Linguistics.

Ruihong Huang and Ellen Riloff. 2012. Bootstrapped training of event extraction classifiers. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 286–295, Avignon, France, April. Association for Computational Linguistics.

Joel Lang and Mirella Lapata. 2011. Unsupervised semantic role induction via split-merge clustering. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1117–1126, Portland, Oregon, USA, June. Association for Computational Linguistics.

Percy Liang and Dan Klein. 2009. Online EM for unsupervised models. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 611–619, Boulder, Colorado, June. Association for Computational Linguistics.

Marvin Minsky. 1974. A framework for representing knowledge. Technical report, Cambridge, MA, USA.

1992. *Proceedings of the Fourth Message Understanding Conference (MUC-4)*. Morgan Kaufmann.

Ani Nenkova and Rebecca Passonneau. 2004. Evaluating content selection in summarization: The pyramid method. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004*, volume 2004, pages 145–152.

Karolina Owczarzak and Hoa T. Dang. 2010. TAC 2010 guided summarization task guidelines.

Siddharth Patwardhan and Ellen Riloff. 2007. Effective information extraction with semantic affinity patterns and relevant regions. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 717–727, Prague, Czech Republic, June. Association for Computational Linguistics.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*.

Hoifung Poon and Pedro Domingos. 2009. Unsupervised semantic parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1–10.

Stephen D. Richardson, William B. Dolan, and Lucy Vanderwende. 1998. MindNet: Acquiring and structuring semantic information from text. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 2*, pages 1098–1102, Montreal, Quebec, Canada, August. Association for Computational Linguistics.

David Rumelhart, 1975. *Notes on a schema for stories*, pages 211–236. Academic Press, Inc.

Roger C. Schank and Robert P. Abelson. 1977. *Scripts, Plans, Goals, and Understanding: An Inquiry Into Human Knowledge Structures*. Lawrence Erlbaum, July.

Yusuke Shinyama and Satoshi Sekine. 2006. Preemptive information extraction using unrestricted relation discovery. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, New York City, USA, June. Association for Computational Linguistics.

Robert S. Swier and Suzanne Stevenson. 2004. Unsupervised semantic role labelling. In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*, pages 95–102, Barcelona, Spain, July. Association for Computational Linguistics.

Michael White, Tanya Korelsky, Claire Cardie, Vincent Ng, David Pierce, and Kiri Wagstaff. 2001. Multidocument summarization via information extraction. In *Proceedings of the First International Conference on Human Language Technology Research*. Association for Computational Linguistics.