Interactions between Narrative Schemas and Document Categories

Dan Simonson
Department of Linguistics
Georgetown University
Washington, DC, 20037, USA
des62@georgetown.edu

Anthony R. Davis
Enterra Solutions
Silver Spring, MD, 20910, USA
tonydavis0@gmail.com

Abstract

The unsupervised extraction of *narrative schemas*—sets of events with associated argument chains—has been explored and evaluated from many angles (Chambers and Jurafsky, 2009; Jans et al. 2012; Balasubramanian et al., 2013; Pichotta and Mooney 2014). While the extraction process and evaluation of the products has been well-researched and debated, little insight has been garnered on properties of narrative schemas themselves. We examine how well extracted narrative schemas align with existing document categories using a novel procedure for retrieving candidate category alignments. This was tested against alternative baseline alignment procedures that disregard some of the complex information the schemas contain. We find that a classifier built with all available information in a schema is more precise than a classifier built with simpler subcomponents. Coreference information plays an crucial role in schematic knowledge.

1 Introduction

In this work, we examine the properties of narrative schemas—sets of events linked by common participants. Though they've been widely investigated, little work has been done to deploy schemas as a component of a larger NLP task, aside from tasks devised purely for validating schemas. To understand what tasks are best suitable for narrative schemas, we've begun to look closely at their properties with the aim of applying them to other NLP tasks.

Intuitively, narrative schemas are plausibly and implicitly linked to the notion of a document category—that is, a schema can represent the narrative commonalities shared by a set of documents. In this work, we set out to try to substantiate this claim in two different ways: we investigate the relationship between schemas and topics and we attempt to use these distributions to classify a set of documents. In Section (2), we describe the variety of techniques that have been attempted to create schemas. In Section (3), we describe the selection criteria for our source data. In Section (4), we discuss our schema extraction procedure, mostly derived from prior work with a few variations. In Section (5), we discuss how categories are assigned to schemas. In Section (6), we outline our different baseline and classifier experiments, and in Sections (7) and (8), we present the results of our experiments. In Sections (9) and (10), we wrap up with implications of these results for future work.

2 Background

What are referred to as *schemas*, *templates*, or *frames* were first introduced in Schank and Abelson (1977) as a generalization of recurring event knowledge. They present scripts as a theory of human memory—events that occur enough are generalized into a script by some aspect of the human mind.

Chambers and Jurafsky (2008; 2009) developed and implemented techniques for the automatic extraction of schemas. A number of papers presenting alternatives, innovations, and variants have followed. Some use co-referent argument pairs—a combination of coreference and syntactic parses to obtain counts of verb-dependency pairs that share a coreferent (Chambers and Jurafsky, 2008;

Chambers and Jurafsky, 2009; Chambers, 2013; Jans et al. 2012; Pichotta and Mooney 2014). Others focus on how the information is presented in a given text, eschewing coreference information altogether to build schemas based on its structure alone (Cheung, Poon, and Vandervende 2013; Balasubramanian et al., 2013). These schemas contain knowledge not of which actors are likely to participate in which actions but of which events are like to occur before and after one another in prose.

In addition to a choice between textual or coreference information providing the basis for scoring, the interactions between different role slots across verbs are handled in roughly two different ways. One approach is to train on individual verb-dependency pairs, themselves associating arguments to verbs (Chambers and Jurafsky, 2008; Chambers and Jurafsky, 2009; **?**; **?**). On the other hand, all role fillers can be handled together as one tuple that acts as the argument to a verb (Pichotta and Mooney 2014; **?**). The key difference is that the verb-dependency approach accepts arguments to a particular verb without giving those arguments any information about the others; the tuple approach informs the arguments about one another in some way. Verb-dependency approaches are more Davidsonian in the degree of freedom given to verb arguments than their tuple-bound counterparts (Davidson 1967).

Candidate insertions into a schema are ranked in different ways. Pointwise mutual information (pmi) is used in a number of approaches (Chambers and Jurafsky, 2008; Chambers and Jurafsky, 2009; Jans et al. 2012) or maximization of probability given features under consideration, including argument types and verb collocations themselves (Jans et al. 2012; Pichotta and Mooney 2014). Balasubramanian et al. (2013) use a graph-ranking algorithm to generate schemas. Some newer work takes a more theoretically sophisticated approach, employing a formal probabilistic model along with a Hidden Markov Model to induce schematic knowledge (Chambers, 2013; Cheung, Poon, and Vandervende 2013).

Most implementations have been evaluated using the narrative cloze task (Chambers and Jurafsky, 2008; Chambers and Jurafsky, 2009; Jans et al. 2012; Pichotta and Mooney 2014). In this procedure, a random verb is removed from a document and the previously extracted schematic knowledge is used to rank alternative verbs that could fill the empty event slot. Balasubramanian et al. (2013)—contrary to other approaches—use human intuitions from Amazon Mechanical Turk to evaluate their schemas.

3 Data Selection

Our data came entirely from the New York Times Corpus (Sandhaus 2008), which consists of around 1.8 million documents from the eponymous newspaper. Each document comes tagged with associated metadata, including date, two types of document categories, tags of people mentioned in each document, and other information.

From the original 1.8 million documents, 38832 were retained to generate schemas after our selection process, described next.

3.1 Keyword and Year Selection

All documents containing the keyword "police" in any form were extracted from the New York Times Corpus. Documents from late 1994 to mid 2008 were retained. This reduced the set to roughly 8% of the original corpus size.

3.2 Categorical Selection

Documents in the NYT corpus are tagged with an `online_producer` property that provides categorical labels for documents. A subset of these categories was then retained, with the intention of providing not only a variety of narratives, but also some more potentially complex distinctions that could be difficult to disentangle. Collectively, this represents a set of documents that are more likely to refer to police as the focus—"noise" and "demonstrations and riots"—than many of those excluded—"international relations" and "United States Armament and Defense." No categories outside of this set were explicitly excluded, however, and nothing prevents these categories from overlapping, which they often do. Most extreme in this regard is the category "Serial Murders", where every article is also contained in "Murders and Attempted Murders."

In total, 38832 documents remain in the corpus of source data. Table (1) lists the categories and gives a breakdown of the distribution of documents across categories.

3.3 Coreference and Dependency Preparation

Documents were parsed and their coreference chains were extracted with Stanford CoreNLP version 3.4.1 (Manning et al. 2014), particularly the Stanford Parser (de Marneffe, MacCartney, and Manning 2006) and the Stanford Deterministic Coreference Resolution System (Lee et al. 2013). From the parser, we used the `collapsed-ccprocessed-dependencies`. We only looked at dependencies related to the verb, and each dependency was collapsed into an appropriate super-category: `agent`, `subj`, `nsubj`, `csubj`, `xsubj` are all mapped to `SUBJ`; `comp`, `obj`, `dobj`, `nsubjpass` to `OBJ`; `iobj` and `prep_.*` to `PREP`.[1]

4 Extracting Schemas

In this section, we discuss in detail two components of how we created schemas. The first is how we scored candidate events for adding to a particular schema, with our score being largely derived from Chambers and Jurafsky (2009). In the second, we discuss how this score is used to generate schemas.

4.1 Scoring Candidate Events

We largely followed Chambers and Jurafsky (2009) in scoring candidate events with respect to a particular schema.

Their score is based on *pmi*, defined in this context as:

$$pmi(\langle w, d\rangle, \langle v, g\rangle) = \log \frac{P(\langle w, d\rangle, \langle v, g\rangle)}{P(\langle w, d\rangle)P(\langle v, g\rangle)} \tag{1}$$

where w and v are verbs, d and g are dependencies. The probabilities P of pairs of narrative events are defined as:

$$P(\langle w, d\rangle, \langle v, g\rangle) = \frac{C(\langle w, d\rangle, \langle v, g\rangle)}{\sum_{x,y}\sum_{d,f} C(\langle w, d\rangle, \langle v, g\rangle)} \tag{2}$$

where $C(\langle w, d\rangle, \langle v, g\rangle)$ is the number of times a co-reference chain contains some word that has d dependency with verb w and some word that has a g dependency with verb v. For example, the pair of sentences "John$_i$ danced poorly. The crowd booed at him$_i$" would contribute one count to $C(\langle$`dance, SUBJ`$\rangle, \langle$`boo, PREP`$\rangle)$.

To include the effect of typed arguments, (Chambers and Jurafsky, 2009) defines sim as:

$$sim(\langle e, d\rangle, \langle e', d'\rangle, a) = pmi(\langle e, d\rangle, \langle e', d'\rangle) + \lambda \log freq(\langle e, d\rangle, \langle e', d'\rangle, a) \tag{3}$$

a represents a specific argument type. $freq(b, b', a)$ returns the corpus count of a filling both b and b'.

Chambers and Jurafsky (2009) used an open set of noun phrase heads to generate their types. Instead, we created an explicit list of preferred types from the top 300 tokens contained in noun phrases. We then removed cardinal numbers from this candidate list, leaving 294 preferred argument types. This was done for two reasons: to reduce data sparsity and to improve performance since $chainsim'$ maximizes over all possible types.

If none of the preferred types are available inside any of the noun phrases of a co-reference chain, the results from the Stanford NER (Finkel, Grenager, and Manning 2005) are checked. After this, any pronouns are used to map a coreference chain to an appropriate fall-back type, either `SELF`, `PERSON`, `THING` or `PEOPLE` as appropriate. If there is no obtainable type, a final fall-back called `THINGY` is used.

Chambers and Jurafsky (2009) point out that sim biases the selection of verbs in favor of adding a new verb that simply shares an argument type with another verb already in the schema. However, this does not guarantee that the type works for all events already in the schema. For this reason, $score$ is defined as follows, to sum over sim values with all current elements of the schema:

$$score(C, a) = \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} sim(\langle e_i, d_i\rangle, \langle e_j, d_j\rangle, a) \tag{4}$$

With sim and $score$, $chainsim'$ is defined as:

$$chainsim'(C, \langle f, g\rangle) = \max_a \left(score(C, a) + \sum_{i=1}^{n} sim(\langle e_i, d_i\rangle, \langle f, g\rangle, a) \right) \tag{5}$$

$chainsim'$ superpositions the influence of two forces on introducing a new pair $\langle f, g\rangle$ to a chain:

[1] Chambers and Jurafsky (2009) include *prep* as one of their argument slots but do not include it in their diagrams: "An *event slot* is a tuple of an event and a particular argument slot (grammatical relation), represented as a pair $\langle v, d\rangle$ where v is a verb and $d \in \{subject, object, prep\}$."

how well $\langle f, g \rangle$ fits in the chain—which constitutes $\sum sim(...)$—and how well the argument a fits within the context of the rest of the chain—the effect of the $score(C, a)$ component. $chainsim'$ finds the best argument for inducing this combination.

Differing from Chambers and Jurafsky, the candidate verb argument type a that maximized $score$ in Formula (5) is also retained to add to the list of types associated with that chain in the schema. If a role slot fails to score higher than a threshold for any existing chains in the schema, a new, un-filled singleton chain is started. If no evidence for a slot was observed in the data with respect to a particular verb, that slot is never considered for addition to any chains associated with that verb.

4.2 Schema Induction Procedure

In this section, we describe criteria for limiting schema growth based on a competition model among schemas for verbs. Chambers and Jurafsky (2009) descend the list of verbs ranked by their $narsim$ score, adding each new verb incrementally with $narsim(N, v_j) > \beta$—creating a new schema if $narsim(N, v_j) < \beta$—or before a hard limit of between six and twelve total events in a schema, a number that varies for different experimental purposes. Given that this algorithm is greedy, it is not entirely clear that it generates schemas that are globally optimal and best represent the narratives exhibited in the corpus.

Our aim is to avoid the creation of "low quality" schemas resulting from the addition of verbs that do not fit particularly well into one schema as compared to others. Yangarber (2003) provides a useful analogy in his description of *counter-training* in the discovery of patterns for information extraction. He notes that an "unsupervised algorithm does not know when to stop learning", so that "in the absence of a good stopping criterion, the resulting list of patterns must be manually reviewed". Yangarber's algorithm relies on competition among several different learners, each seeking patterns for a different "scenario" (a topic or domain). A pattern might have evidence favoring a learner to select it, but if learners for other scenarios also find evidence to acquire it, that counts against the first learners evidence.

The analogy that carries over to narrative schemas is that they reflect topics or domains, like Yangarber's scenarios. Narrative schemas are instantiated in individual documents, as sets of clauses. Thus, a particular clause should "belong to a single schema". On this analogy, we can formulate a version of counter-training by having each schema compete for the elements that constitute it. Those elements are verbs, which are thus the analogs of patterns. Their individual instantiations are clauses in documents – that is, a verb, its dependencies and their fillers. Clauses are thus the analogs to documents, because we wish to determine, for a given clause, which schema it instantiates, if any.

Algorithm 1: Counter-training for narrative event chain construction.

```
Data: Seed schemas, a scoring function
      scoring, pruning conditions
Result: narrative schemas
while number of SchemasGrowing and
Candidates both > 1 do
    initialize simtables S
    for every schema ∈ SchemasGrowing do
        initialize simtable s
        for every candidate ∈ Candidates do
            add
            scoring(schema, candidate) to
            s
        add s to S
    broadness[can] = Σ_{s∈S} Σ_{c∈s} 1
    for simtable in simtables do
        for can in broadness do
            simtable[can] −=
            broadness[can]
    induct highest-ranked Candidates into
    SchemasGrowing
    prune SchemasGrowing and
    Candidates
return GrownSchemas
```

Each schema ranks potential new additions in competition with other schemas. The specific process for this is detailed in Algorithm (1). In short, every candidate event is scored with respect to each schema and saved in $simtable$. Then the $broadness$—how well each candidate event scored with respect to all schemas—is computed. Each score is penalized based on the broadness, and the highest-ranked candidates are inducted into their respective schemas. The list of schemas and candidates are pruned according to the provided rules, and the process continues while there are both still candidates and schemas available.

Using a broadness table allows for schemas to compete with one another, and to do so irrespective of the order they are in. If many competing schemas rank a candidate event highly, they may only add it to themselves if the score outweighs the allotted penalties. If too many instances of a verb and its dependents seem to fit in different schemas, we drop it from the list of candidate additions to our narrative schemas. This does not preclude a verb belonging to two or more narrative schemas, since its individual occurrences might unmistakably belong to one schema or another, even after penalties have been deducted.

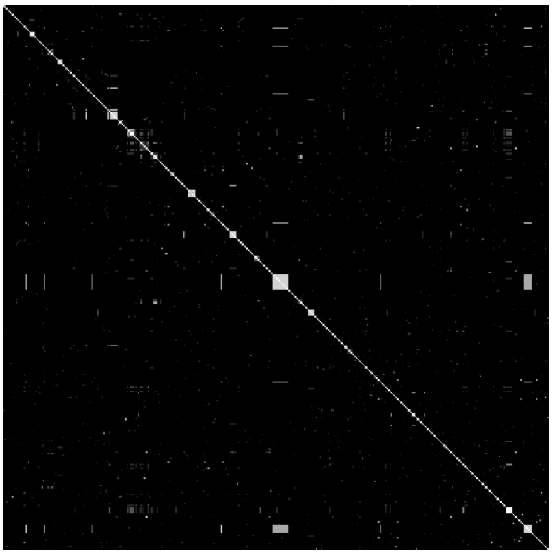

Figure 1: A grayscale confusion matrix showing overlap of events in schemas. Each column and row of pixels represents a schema, the schemas themselves arranged orthographically. Increasing brightness in a particular row and column indicates that more events overlap between the schemas represented by those respective rows and columns. Our counter-training algorithm is intended to produce schemas that are unique from others—that is, that follow the diagonal strongly.

Empirical evaluation with the cloze task is forthcoming. While we cannot enumerate all 800 of our schemas here,[2] Figure (2) and (3) show examples that indicate that our schemas are at least comparable with those others have extracted and are sufficient for looking at the interaction between schemas and document categories.

Our algorithm allows for the generation of duplicate schemas. Two schemas can easily converge

[2]The full set can be found, in multiple formats, at: `http://schemas.thedansimonson.com`

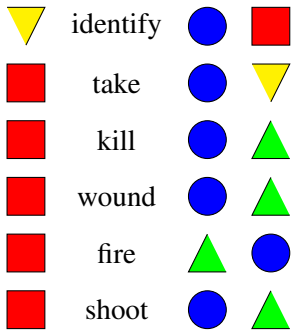

Figure 2: A schema extracted using our technique, generated for and used in the classification task. The red square and blue circle both indicate different `PERSON`s. The downward pointing yellow triangle indicates some `THINGY`; the upward pointing green triangle indicates either `baghdad` or a `THINGY`.

if they were seeded with verbs that were closely related; once they include the same events, they are effectively identical. Figure (1) shows overlap between all 800 schemas.

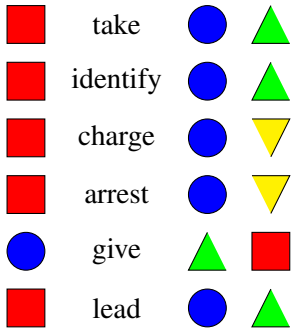

Figure 3: Another schema extracted using our technique, generated for and used in the classification task. Red squares are a `police` chain. Both blue circles and green, upward pointing triangles are independent `PERSON` chains. Downward pointing triangles are a chain referring to a `killing`.

5 Preparing Schemas for Classification Experiments

To better understand the properties of schemas, we will investigate how well schemas correlate with the document categories assigned within the NYT corpus. We will look at the schemas in two different ways—first, by assigning document categories to schemas, then by using these assignments to complete a categorization task. We do not ex-

pect the system to perform better than proven categorization techniques—rather, the categorization task acts as a proxy for investigating the distributional properties of schemas.

5.1 Retrieving Category Counts for Schemas

To employ schemas for classification, we will interpret them as a set of features. Effectively, if we think of the different event argument slots as nodes of a graph, the chains can be thought of as edges between nodes. These edges are pairs of verb dependency pairs which we will refer to as *co-referring argument pairs* (or CAPs, for short). To a great extent, CAPs preserve the information in the schema—the shared role fillers between events—while allowing for partial matches.

For example, Figure (2) contains a number of different chains. Some CAPs derived from this schema are $\{\langle$kill, SUBJ$\rangle, \langle$shoot, SUBJ$\rangle\}$ from the red square PERSON chain—derived, intuitively, from the fact that someone who shoots often kills—$\{\langle$fire, PREP$\rangle, \langle$shoot, OBJ$\rangle\}$ from the blue circle PERSON chain—derived from the fact that one may "shoot someone," but also "fire *at* someone"—among many, many others. This schema alone contains 37 CAPs: 15 each from the two chains that are shared in each and every role slot, and 7 from the other two auxiliary slots.

For a given set of chains S^C from schema S, we disentangle the CAPs contained via the following:

$$\text{CAPs}(S) = \{\{vd_a, vd_b\} : \bigwedge_{x \in \{a,b\}} vd_x \in C \in S^C\} \tag{6}$$

where C is a chain contained in the set of chains S^C, and vd_x is any verb-dependency pair; a and b are arbitrary indices. We then can assign weights to a category c for a schema S by counting the categories of the documents that each CAP appears in, or more specifically:

$$W(c, S) = \sum_{d \in D} \begin{cases} w(c): & d \cap \text{CAPs}(S) \neq \emptyset \\ 0: & otherwise \end{cases} \tag{7}$$

where D is the set of sets of CAPs from each of our training documents. $w(c)$ is a weighting function for a category. If we are working with simple document counts, $w_1(c) = 1$ is sufficient; alternatively, a cf-idf—like tf-idf but with categories instead of terms—could be used. This measure uses $w_{idf}(c) = \frac{N}{n_c}$, where N is the total number of documents in the corpus and n_c is the number of documents denoted as class c.

6 Classification Experiments

In order to understand the extent to which schematic information interacts with document categories, we considered individual, plausible components of schemas as baselines to compare against the performance of our full blown schema-based classifier. We discuss these in this section, as well as how the classification was performed, and how the target data set was chosen.

Each experiment represents a different way of extracting features from each schema. In other words, we still begin with schemas, but we extract the features between experiments. Each technique is intended as a plausible candidate for explaining how our schematic classifier works, working from the simplest to more complex collocations.

6.1 Experimental Models

In this section, we will discuss each of our baseline models, leading up to the features discussed in Section (5.1).

6.1.1 Bag of Words Model

The bag of words model used here relies only on the presence of events found in our schemas for classification. Instead of thinking of each schema as a set of chains that are decomposed into CAPs, we look at each schema as a set of events S^E:

$$\mathbb{W}(S) = \{v_x : v_x \in S^E\} \tag{8}$$

where v_x is a verb and x is an arbitrary integer. The $\mathbb{W}$ of the schema in Figure (2) is $\{$shoot, fire, wound, kill, take, identify$\}$.

6.1.2 Document Co-presence Model

In the document co-presence baseline model, if two events both appear in a document—regardless of their location or anything else—then that counts as an instance of that feature.

$$\mathbb{D}(S) = \{\{v_a, v_b\} : \bigwedge_{x \in \{a,b\}} v_x \in S^E\} \tag{9}$$

All permutations of pairs of events are considered. In a schema of size 6, this means that there are 15 pairs of events as features: $\{\{$shoot, fire$\}, \{$shoot, wound$\}, \ldots$ *etc.*$\}$.

6.1.3 Coreference Co-presence Model

Our final baseline creates pairs any two events which share co-referrent arguments. We do not

include the specific argument slot. Now using S^C, the set of chains from schema S, instead of S^E:

$$\mathbb{C}(S) = \{\{v_a, v_b\} : \bigwedge_{x \in \{a,b\}} v_x \in S^C\} \quad (10)$$

This model's features are nearly schematic in nature, except that the features lack the specific slot wherein co-presence was defined; at this point, we effectively are using schemas without their role slot labels. Features derived from the schema in Figure (2) are no different from the last baseline because all events are shared with at least one chain. However, the interpretation of our hold-out documents changes. Because we are now looking at coreference, it is not the mere presence of a pair of events in the text, but their linkage through their arguments via coreference that counts.

6.1.4 Schematic Classifier

This is our schematic classifier, as discussed above and illustrated with Equation (6). Note that Equation (10) is nearly identical to Equation (6); v has been swapped with vd representing the set of verb-dependency pairs. With verb-dependency pairs instead of verbs alone, we have built-up to a set of features that closely approximates our schemas.

6.2 Implementation

We used the `scikit-learn` class `sklearn.naive_bayes.MultinomialNB` to classify our documents (Pedregosa et al. 2011). Because our document categories overlap, we took a one-vs-all classification strategy for each document class; each document category represents a split into + or − classes. For the classification task, to give as much information as possible to the classifier, we generated 800 schemas seeded with the 800 most frequent verbs. We held-out $1/10^{th}$ of documents for evaluation.

In performing classification, we conducted a "rank descent." We started with the highest weighted category for a given feature in our first test, then used the two highest-weighted categories in the second experiment, etc., until every category that appeared with the feature is applied.

We completed the classification task in two separate sets of experiments using the raw counts weighting (w_1) in one and the cf-idf (w_{idf}) weighting scheme in the other.

7 Results

Table (1) contains a breakdown by category of peak performance. Categories that were better represented tend to have higher peak F1 scores. More poorly represented categories tended to peak in performance with the CAPs or at least coreference information provided by the coreference co-presence model $\mathbb{C}$, though this was not entirely the case—the very frequent category "crime" peaked with the $\mathbb{C}$.

Table 1: Number of documents per category retained from the "police" subset, along with the rank n at which the rank descent reached the peak F1 value, which of the weighting functions w_x—w_1 or w_{idf}—was used from Section (5.1) and which of the models was used from Section (6.1) for which performance peaked with respect to F1. $\mathbb{W}$ is the bag of words model, $\mathbb{D}$ is document co-presence, $\mathbb{C}$ is coreference co-presence, and CAPs represents a fully schematic classifier. N is the number of documents in a respective category. Some category names have been shortened or abbreviated.

Category	N	F1	n	w_x	Model
Terrorism	16,290	0.422	9	*idf*	$\mathbb{W}$
Crime	14,685	0.461	6	*idf*	$\mathbb{C}$
Murders	13,872	0.430	1	1	$\mathbb{W}$
World Trade Ctr.	8,916	0.213	3	1	CAPs
Violence	6,450	0.183	5	*idf*	$\mathbb{C}$
Demonstr. and Riots	6,430	0.193	4	*idf*	$\mathbb{W}$
Accidents	5,719	0.166	4	1	$\mathbb{W}$
Police Brutality	4,627	0.237	2	1	$\mathbb{W}$
Blacks	3,522	0.166	6	*idf*	$\mathbb{D}$
Law and Legislation	3,319	0.321	2	1	$\mathbb{W}$
Frauds	1,848	0.136	7	*idf*	$\mathbb{D}$
Attacks on Police	1,621	0.168	3	1	$\mathbb{C}$
Organized Crime	871	0.098	4	*idf*	$\mathbb{C}$
Serial Murders	918	0.075	8	1	CAPs
Cocaine	464	0.061	5	*idf*	CAPs
Suburbs	303	0.108	3	*idf*	CAPs
Noise	206	0.037	14	1	$\mathbb{D}$
Prison Escapes	137	0.100	2	*idf*	CAPs

Figures (4) and (5) illustrate precision-recall curves for both series of *up to rank n* experiments. In all cases, n goes up as we move from left to right; recall increases with each increase in n.

8 Discussion

Remarkably, we see some capability for schema-specific features to classify documents despite being generated without any explicit knowledge of the classifications they denote. Not in all cases is this the best, but it tends to help bolster performance in under-represented categories within the corpus. The precision-recall curves in Figures (4) and (5) illustrate our point—as we remove features

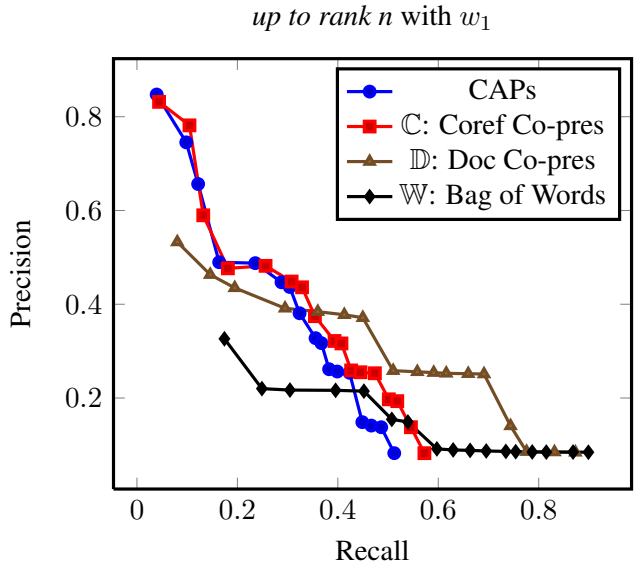

Figure 4: Precision/Recall curves for the *up to rank n* classification experiment using w_1 to assign categories to schemas.

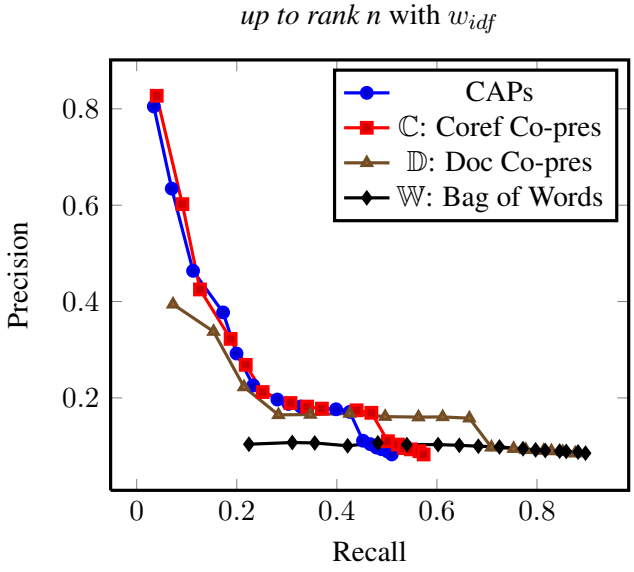

Figure 5: Precision/Recall curves for the *up to rank n* classification experiment using w_{idf} to attach category assignments to schemas.

that our schemas uniquely provide, the peak precision generally declines. This shows that the features included in schemas do possess information specific to their associated document categories.

Of course, the rather simplified classifiers we've presented are by no means reflective of an industry standard classifier.[3] The number of features—only 6901 unique CAPs available, 1629 word types in the 𝕎 baseline—is less than what would be available to a typical bag of words analysis on the same data set—193702 word types. This performance produces precision-recall curves with a concave shape. However, what we do see is a suitable illustration that, with respect to the relationship between schemas and categories, the whole is greater than the sum of its parts.

Also worth noting is the fact that the precision-recall curve of the schematic classifier and the coreference co-presence classifier ℂ nearly adhere to one another. Figure (2) gives a great example of why slot information may not be helpful in all circumstances. In this schema, there are two very clear individuals in most of the events: a shooter of some sort, and someone who was shot. What about with *identify* and *take*? These are a bit more ambiguous; the precise utility of each exact argument slot is not as clear. The connections created through coreference, however, remain quite relevant and, alone, less error prone. This puts into question approaches that leave out coreference (Cheung, Poon, and Vandervende 2013; Balasubramanian et al., 2013)—with respect to this task, something was lost without it.

It is also necessary to critically question the efficacy of our source data, especially the largely unknown criteria used by the NYT Indexing Service to determine document categories. With respect to the schema in Figure (2), most individuals indubitably would say that such a schema is associated with murder. However, there are plenty of examples where *shooting*, *wounding*, and *killing* are not classified by the NYT Indexing Service as "Murders and Attempted Murders:"

> "A Brooklyn grand jury has cleared two police officers in the killing of an unarmed man whom they shot 18 times..."

[3]While our F1 scores across categories averaged 0.199, a non-schematic, bag-of-words Naïve Bayes classifier using all available word types averaged 0.458. Most categories outperformed the non-schematic classifier, except for Suburbs and Prison Escapes, which scored 0.000 with the non-schematic classifier.

> "The United States Marshal who shot and wounded a Queens high school student Thursday after mistaking the candy bar he was holding for a revolver..."
>
> "...the Police Department is being scrutinized over the shooting of several civilians by officers... a Hispanic teen-ager was shot in the back last month in Washington Heights."

In the words of Joe Strummer, "murder is a crime, unless it is done by a policeman." While we did not apply the types of role fillers explicitly to the classification task, these sorts of "errors" motivate the use of role fillers in future work.

9 Conclusions

We have shown techniques for deriving features from narrative schemas, and shown that features derived from narrative schemas are more than the sum of their parts. In particular, coreference information is a crucial component of them and seems—of the set of interpretations of schemas used—to produce the most substantial boost in precision.

10 Future Work

The long term goal of this work is to apply the information contained in narrative schemas to a real-world application. Knowing that schemas can act as precise identifiers of document categories improves our confidence in their usefulness. We hope to experiment with the use of additional features so that narrative schemas can serve as the basis for richer unsupervised knowledge extraction. We have discussed preliminary ideas for new ways to generate schemas as well, which we soon hope to evaluate.

Acknowledgments

We'd like to thank the Georgetown Department of Linguistics for continued support, Amir Zeldes and Nate Chambers for feedback and discussions on components of this work, and the peer reviewers for insightful critiques.

References

Balasubramanian, N., Soderland, S., Mausam, & Etzioni, O. 2013. Generating Coherent Event Schemas at Scale. In *EMNLP* (pp. 1721-1731).

Chambers, N., & Jurafsky, D. 2008. Unsupervised Learning of Narrative Event Chains. In *ACL* (pp. 789-797).

Chambers, N., & Jurafsky, D. 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: Volume 2-Volume 2* (pp. 602-610). Association for Computational Linguistics. Chicago

Chambers, N., & Jurafsky, D. 2011. Template-based information extraction without the templates. In *ACL-HLT 2011* (pp. 976-986). Association for Computational Linguistics.

Chambers, N. 2013. Event Schema Induction with a Probabilistic Entity-Driven Model. In *EMNLP* (pp. 1797-1807).

Cheung, J. C. K., Poon, H., & Vanderwende, L. 2013. Probabilistic frame induction. In *NAACL-HLT 2013* Association for Computational Linguistics.

Davidson, D. 1967. In Nicholas Rescher (ed.), *The Logic of Decision and Action*. University of Pittsburgh Press.

de Marneffe, M., MacCartney, B., and Manning, C.D. 2006. Generating Typed Dependency Parses from Phrase Structure Parses. In *LREC 2006*.

Finkel, J.R., Grenager, T., and Manning, C. 2005. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. In *ACL 2005* (pp. 363-370).

Jans, B., Bethard, S., Vuli, I., & Moens, M. F. 2012. Skip n-grams and ranking functions for predicting script events. In *EACL* (pp. 336-344). Association for Computational Linguistics.

Lee, H., Chang, A., Peirsman, Y., Chambers, N., Surdeanu, M., and Jurafsky, D. 2013. Deterministic coreference resolution based on entity-centric, precision-ranked rules. *Computational Linguistics* 39(4).

Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S. J., and McClosky, D. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *ACL* (System Demonstrations, pp. 55-60).

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, E. 2011. Scikit-learn: Machine learning in Python. The Journal of Machine Learning Research, 12, (pp. 2825-2830).

Pichotta, K., & Mooney, R. J. 2014. Statistical Script Learning with Multi-Argument Events. In *EACL* (pp. 220-229).

Sandhaus, E. 2008. *The New York Times Annotated Corpus*. Linguistic Data Consortium, Philadelphia.

Schank, R.C. & Abelson, R.P. 1977. Scripts, plans, goals and understanding. Lawrence Erlbaum.

Yangarber, R. 2003. Counter-training in discovery of semantic patterns. In *ACL* (pp. 343-350). Association for Computational Linguistics.