ISSN 1392–124X (print), ISSN 2335–884X (online) INFORMATION TECHNOLOGY AND CONTROL, 2014, T. 43, Nr. 4

Constructing Domain Templates with Concept Hierarchy as Background Knowledge

Mitja Trampuš

Jozef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia
e-mail: mitja.trampus@ijs.si

Dunja Mladenić

Jozef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia
e-mail: dunja.mladenic@ijs.si

crossref http://dx.doi.org/10.5755/j01.itc.43.4.6899

Abstract. In recent years, both academia and the industry have seen a push for converting unstructured data, most commonly text, into structured representations. A relatively poorly explored challenge in this area is that of domain template construction: for a domain, we wish to find the attributes with which texts from that domain can be meaningfully represented. For example, given the domain of news reports on bombing attacks, we would like to identify the existence of concepts like "victim" and "perpetrator". We introduce two new methods for this task, both operating on semantic representations of input data and exploiting the hierarchical organization of features, something not explored in prior art. We evaluate on multiple datasets/domains and achieve performance at least comparable to a state of the art method on a set of "real world" scenarios while additionally identifying fine-grained type information for properties: for example, the bombing attack victim is found to be of type "defender" (policeman, guard, ...).We also provide the first fully documented evaluation methodology, publicly available labeled datasets and golden standard outputs for this research problem, supporting and facilitating future work in the area.

Keywords: text mining; open-domain information extraction; schema induction; graph mining.

1. Introduction

One of the long-standing goals of AI is to convert natural language text into structured representation(s): from linguistic parsing to extracting entities, relations, tabular forms, and ultimately, expressing documents as a series of logic statements. Such structures provide meta-information, describing the role of and relations between individual words, sentence fragments or larger parts of texts. Therefore, structured representations are less ambiguous and inherently easier to search and navigate.

So far, reasonably reliable methods have been developed for structuring text by annotating and identifying a specific subset of information, most commonly named entities. We tackle a related but distinctly different problem that arises beforehand: suggesting the *structure* with which the text should be represented, the *types* of information that should be extracted.

Closely related to attempts at structuring unstructured text, recent years have seen a proliferation of semantic methods and representations. Prominent examples include Google's structured search results for movies, albums, people, cities and more; Facebook's structured search interface (e.g. "men from *C* who are friends with P_1 and P_2 and are older than *Y* years"); and the Wikidata project [36], a systematic push by Wikimedia to semanticize the semi-structured information in Wikipedia's infoboxes.

Having data that is both structured and semantic, i.e. represented using concepts from a knowledge base, further decreases ambiguity and allows for easier linking with other knowledge. The latter is particularly important when dealing with natural language: humans assume a lot of shared context ("common sense") when speaking or writing, and combining raw data with background knowledge in the form of a knowledge base (KB) or an ontology can help algorithms compensate for their lack of context [40, 16]. In view of these trends, part of our goal was to develop a method that produces semantic, KB-aligned outputs.

Problem statement. We are given a set of documents from a single, relatively restricted domain, for example "reports of bombing attacks", "weather reports" or "biographies of renowned physicists." The task is to identify, in an unsupervised manner, the most salient properties that can be defined for most of the given documents; for example, given the "bombing attacks" domain, we wish to detect "attacker", "the destroyed property", "victims" etc. as properties that are pervasively present in those articles. We tentatively define salient properties as those that would allow a human, if she were given only the values of those properties for an unseen document, to produce as good an abstract of the *unseen* document as possible. The properties will be described by their prevailing context and will be assigned a type. For example, the "attacker" property from the previous sentence might be output as $\boxed{person} \xrightarrow{\text{detonate}} \boxed{\text{bomb}}$. Here, *person* is the type while $\xrightarrow{\text{detonate}}$ and $\boxed{\text{bomb}}$ provide sufficient context to determine this *person* is the attacker. Automatically assigning the label "attacker" to this property is beyond the scope of this (and related) work. We call the collection of these properties for a specific topic a *domain template* or *topic template*.

Motivation. While the output of domain template construction methods contains some noise and has to be checked by humans, it has the potential of greatly reducing human involvement and effort in tasks that require insight into the structure of a domain.

Semi-automatic ontology extension is one such use case. Existing relation extraction methods are sometimes used to extend the lowest, fact-based levels of ontologies (e.g. adding `bornIn` relations between persons and places). Templates, on the other hand, provide input for extending the middle level of ontologies: when introducing a new abstract concept *C* (e.g. "football player") to the ontology, a topic template derived from documents on *C* can suggest properties and relations (e.g. "played for", "goals scored") to be associated with new instances of *C* in the ontology.

In a similar vein, domain templates guide and constrain *Information Extraction* (IE) methods which have a wide variety of applications. Present-day IE algorithms are most often supervised in nature and depend on manual creation of topic templates and training documents with labeled slot fillers. Computer-assisted creation of topic templates thus lowers the entry barrier to using IE. Not only does it provide the templates, a high number of labeled slot fillers is almost always a byproduct of automatic template creation.

Another added value of templates is that they expose the key properties of a text type. This makes them potentially suitable for guiding **summarization** or other text shortening tasks by identifying text fragments that should be scored higher.

In combination with information extraction methods, topic templates allow us to create writing "mentors", automated ways of suggesting **missing content** to be included into a document with a known topic. For example, if the user is posting a sales ad for a car (TV, house, ...) – something most people don't do often – the system could remind her of information that is typically included in such ads but the user's ad lacks. Similarly, a journalist covering a story could be reminded of types of information typically covered in related articles but not in hers. On a larger scale, we can imagine a system that analyzes all Wikipedia articles from a given category, derives the template and identifies pages that are missing some of the "standard" properties (e.g. "of all *German Physicist* pages, only Max Planck's lacks info about his schooling").

Focusing on documents from a single domain might seem restrictive; however, this focus emerges naturally. On the data side, document collections often already include keyword annotations or a topic categorization with which we can obtain single-domain subcollections; alternatively, we show in Section 4.1 how a collection of documents from a domain of interest can be obtained from the internet with minimal human involvement.

In contrast to the so-called open information extraction or other approaches that identify domain templates from non-domain-specific text collections, having a clean single-domain input also allows methods to find richer templates, with roles that would otherwise get drowned in the noise much more easily or take enormous amounts of data to emerge statistically. Most of the related work [13, 32, 14, 28] therefore also assumes a single domain at a time as input.

Contributions. The main contributions of this paper are as follows:

- We are the first to integrate background knowledge into the task of unsupervised construction of domain templates and solve the task in two ways, both using a data representation that is significantly different from the norm.
- We achieve performance at least on par with the state of the art and additionally produce, unlike the work so far, fine-grained type constraints for template slots.
- Evaluation for this task is complicated, and there has so far been no well-documented evaluation methodology or sizable public datasets and golden standards for comparing methods. We provide both.

Structure. The rest of the paper is structured as follows. Section 2 reviews related work. Section 3 describes the two novel methods for creating domain templates, the Frequent Generalized Subgraph method (FGS; Section 3.2 and the Characteristic Triplet method (CT; Section 3.3) as well as the preprocessing common to both. Evaluation is described in Section 4 and Section 5 discusses the results. We finish with conclusions in Section 6.

2. Related work

Topic template construction is related to the more established field of Information Extraction (IE). However, our focus is elsewhere, on the quality of template slots themselves rather than the quality of slot fillers.

In traditional IE, the topic domain is constructed beforehand and remains fixed. Of even more interest to us are therefore Open Information Extraction systems; "open" in the task name refers to the fact that these systems learn new relations (effectively, template slots) on the fly. The first such system was TextRunner [3, 22], which extracts sentence fragments of the form (*entity, verb phrase, entity*), for example (*the Saints, win, the Superbowl*). Balasubramanian et al. [2] recently described a method for clustering these syntactic patterns based on their co-reference frequency and abstracting the entities using WordNet; each cluster then approximates a semantic relation (expressed with several syntactic ones) with typed slots instead of entities. The focus is on commonsense knowledge at the web scale.

Also prominent in the area of open information extraction is NELL, the Never Ending Language Learner [5] which has been continuously scanning the web for several years and is learning to discover new entity types and relations (with high-quality slot fillers as a necessary side effect), producing a "topic template" of common-sense knowledge. In particular, its Coupled Pattern Learner (CPL) component [6] suggests relations based on frequently co-referencing syntactic patterns, not unlike Balasubramanijan et al. above. All open IE systems are based on the same very rough core idea as the topic construction methods listed later on: finding repeating text patterns.

The task of domain template construction itself has seen far less research activity. The majority of existing methods start by representing the documents as dependency parse trees, thus abstracting away some of the language variability and making pattern discovery more feasible. The patterns found in these trees are often further clustered to arrive at more general, semantic patterns or pattern groups. In the remainder of this section, we describe the most closely related contributions in more detail.

Several articles focus on a narrow domain and/or assume a large amount of domain-specific background knowledge. For example, Das et al. [13] analyze weather reports to extract patterns of the form `[weather front type] is moving towards [compass direction]` where they manually create rules (based on shallow semantic parsing roles and part-of-speech tags) for identifying instances of concepts such as `compass direction` and `weather front type`. Once these concepts are identified, they cluster verbs based on WordNet and then construct template patterns for each verb cluster independently; a pattern is every frequent subsequence of semantic roles within sentences involving verbs from the verb cluster. The idea is only partially transferable to the open domain; authors themselves point out that they rely on the formulaic language that is typical of weather reports.

The method by Shinyama and Sekine [32] makes no assumptions about the domain but does limit itself to discovering named-entity slots. It tags named entities and clusters them based on their surrounding context in constituency parse trees. The problem of data sparsity (a logical statement can be expressed with many natural language syntactic trees) is alleviated by simultaneously analyzing multiple news articles about a single news story – an approach also taken by our FGS method in Section 3.2. In the end, each domain slot is described by the set of its common syntactic contexts.

Filatova et al. [14] use a tf-idf-like measure to identify the top 50 verbs for the domain and extract all dependency parse trees in which those verbs appear. The trees are then generalized: every named entity is replaced with its type (person, location, organization, number). Frequent subtree mining is used on these trees to identify all subtrees occurring more than a predetermined number of times. From the frequent trees, all the nodes except the verb and the slot node (i.e. the generalized named entity) are removed; the remainder represents a template slot. The approach is representative in spirit of most of the related work while also being well evaluated, which is why we choose to compare against it. The method is unnamed; because it focuses on modifiers of frequent verbs, we refer to it as the Frequent Verb Modifier (FVM) method.

Chambers and Jurafsky [8] take a different approach: they first cluster verbs based on how closely together they co-occur in documents. For each cluster, they treat cluster verbs' modifiers (object, subject) as slots and further cluster them by representing each verb-modifier pair (e.g. (*explode, subj*)) as a vector of other verb-modifier pairs that tend to refer to the same noun phrase (e.g. [(*plant, obj*), (*injure, subj*)]). Both rounds of clustering observe a number of additional constraints omitted here. The method is also capable of detecting topics from a mixture of documents, positioning the work close to open information extraction. A more recent version by Chambers [7] uses the same features (verb-modifier pairs) but replaces the two clustering rounds with a single graphical model.

Similarly, Cheung et al. [9] suggest another graphical model approach that does not need documents to be topic-labeled in advance and attempts to cluster them at runtime. They do so with a variant of a Hidden Markov Model (HMM). The observed state of the HMM is composed of the verb encountered in the text and its dependents (obtained from the dependency parse tree) along with their type (subject, object). The hidden state consists of the topic and the micro-level event (essentially a verb concept, e.g. "walking"). The probabilities for the hidden state are influenced by the previous hidden state, giving the graphical model an HMM-like structure.

The above two papers [7, 9] represent an interesting and promising attempt at bridging the gap between single-domain template construction and open information extraction. Unfortunately, they develop and test on the MUC-4 dataset, which is a mixture of documents from four preselected domains and less than 50% of non-topical (noise) documents, a very unlikely real-world scenario. Their performance at web scale or on another dataset with a high number of topics or high amount of noise in unknown. Regardless of that, we believe the MUC-4 dataset and the accompanying evaluation methodology to be a poor choice for two reasons. First, MUC-4 only provides 2-4 slots per template, which is too low to sufficiently describe a real-world scenario. Second, the dataset evaluates the performance of an Information Extraction system built on top of the extracted templates, *not* the quality of the templates themselves. This fails to capture the suitability of templates for other purposes, for example ontology extension or summarization.

Finally, Qiu et al. [28] propose a method with more involved preprocessing. Unlike the other methods, which consume parse trees, this method operates on semantic frames coming from a Semantic Role Labeling (SRL) system. Within each document, the frames are connected into a graph based on their argument similarity and proximity in text. The frames across document graphs are clustered with an EM algorithm to identify clusters of frames that semantically likely to represent the same template slot(s). This approach is interesting in that it is markedly different from the others; sadly, there is no quantitative evaluation of the quality of the produced templates and even the qualitative evaluation (= sample outputs) is scarce.

Note that almost all of the related work, like ours, concerns itself with newswire or similar well-written documents, allowing parsers to play a crucial role. For less structured texts, parsing is not feasible any more and domain-specific approaches are needed. This was observed for example by Michelson and Knoblock [24] who automatically construct a domain schema from craigslist ad titles, deriving for example a taxonomy of cars and their attributes. Their templates also significantly differ from all the approaches listed above in that they are not verb- or action-centric.

Our proposed method is unique in that it tightly integrates background knowledge into the template construction process; all existing approaches rely instead on contextual similarities to cluster words or phrases into latent slots. None of the above methods explore the benefits and shortcomings of using semantic background knowledge. However, a hierarchy/ lattice of concepts, the form of background knowledge employed by us, was recently successfully used in related tasks of constructing ontologies from relational databases in a data-centric fashion [31] and semiautomatic ontology building [18]. An approach similar to ours has also been successfully used in a related and equally novel task of event prediction [29]: Starting with events from news titles (e.g. "Tsunami hit Malaysia", "Tornado struck in Indonesia"), the authors employed background knowledge to derive generic events and compute likely causality relations between them, e.g. a "[natural disaster] hit [Asian country]" event predicts a "[number] people die in [Asian country]" event.

A note on terminology. The domain template construction task is young and has unconsolidated terminology. The task has so far been tackled by people coming from different backgrounds, using different names for the task itself and the concepts related to it. We collected the assorted terms in Table 1. Our terminology mostly follows that of Filatova. Qiu's is influenced by the early terminology introduced in the 90s for Information Extraction tasks (where the domain templates were created by hand), e.g. at the Message Understanding Conference (MUC) [15]. Chambers's "roles" and "role fillers" are normally used with Semantic Role Labeling (SRL) [11]; interestingly, he does not use the SRL term "frame" for templates. Shinyama's naming choices are strongly rooted in relational databases. Cheung's reflect their use of a graphical model.

Table 1. Consolidation of terminology in related work. Following our terminology, the *domain* is what the input documents have in common. *Properties/slots* are the concepts we would like to discover. *Slot filler* is a specific value that can fill the slot; this is what algorithms have to abstract away to produce the slots. *Patterns* are the syntactic context of slots using which the algorithm identifies slots and usually also presents them to the user; their content and representation are highly algorithm-specific. The *domain template* is the collection of all patterns for a domain and is the final output of the algorithm.

This article	domain, topic	slot, property	slot filler	pattern, triplet	schema, template
Filatova [14]	domain, topic	slot	slot filler	slot structure	dom. template
Das [13]	domain	slot	slot value	template	—
Chambers [7]	domain	role, slot	role filler	syntactic relation	narrative schema
Qiu [28]	scenario	salient aspect, slot	sample modifier	—	scenario template
Cheung [9]	scenario	argument	(emission)	caseframe	frame
Shinyama [32]	—	relation	—	basic pattern	unrestricted relations
Example	bombing attack	attacker	John Smith	$\text{bomb} \xrightarrow{\text{kill}} person$	(all slots)

3. Methods for obtaining domain templates

We present two methods for unsupervised construction of domain templates based on semantic representation of input documents.

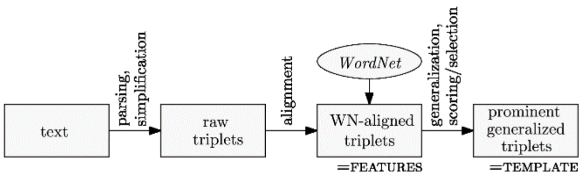


Figure 1. Pipeline sketch for both proposed algorithms. The text is reduced to relational triplets, and they are aligned (linked) to WordNet. In the last and main step, generalizations of these triplets are formed and only the promising ones are retained and presented as the template.

Both methods are based on a graph-like representation of documents and topic templates, with labeled nodes denoting concepts and labeled edges denoting relations between them. The methods follow the pipeline very roughly outlined in Fig. 1.

We use the following **notation** in text:

- $\boxed{\text{Node}}$ for concepts extracted directly from documents, e.g. "Obama".
- $\boxed{NodeType}$ for generic, automatically inferred concepts, e.g. "person".
- $\boxed{\text{Node}_1}\xrightarrow{\text{relation}}\boxed{\text{Node}_2}$ for relations.

The **assumptions** that we make about the **input data** are as follows:

- A collection of plain-text documents from the domain of interest is available.
- The key information in input documents (and the desired output) can be represented with relational triplets (here, $\boxed{subject}\xrightarrow{\text{verb}}\boxed{object}$ or $\boxed{verb}\xrightarrow{\text{dependency}}\boxed{property}$). This assumption is likely to be partially violated, which can be alleviated with input data redundancy.

Both methods share the preprocessing stage in which said triplets are extracted from plain text.

In the second, **main part** of the algorithm, the methods take markedly different approaches. The first, presented in Section 3.2, attempts to discover regularities in the **semantic structure** of the documents, i.e. the entities appearing as well as well as the relations interconnecting them. For example, in documents reporting on murders, we hope to find a recurrent complex structure like

$$\boxed{officer}\xrightarrow{\text{apprehend}}\boxed{person}\begin{array}{l}\xrightarrow{\text{kill}}\boxed{person}\\ \xrightarrow{\text{receive}}\boxed{\text{sentence}}\end{array}.$$

The method assumes such complex semantic structures are extremely unlikely to appear outside the context for which they are characteristic (i.e. *murder* stories) and searches for such structures in a manner reminiscent of frequent itemset mining.

The second approach relaxes the assumption on how common these large semantic structures are and instead looks for individual **topic-characteristic triplets** ($\boxed{officer}\xrightarrow{\text{apprehend}}\boxed{person}$, $\boxed{person}\xrightarrow{\text{kill}}\boxed{person}$ and $\boxed{person}\xrightarrow{\text{receive}}\boxed{sentence}$ separately), which can be seen as a reduction in the size of sought-after semantic structures. As these small structures appear more commonly even outside the target domain (i.e. in non-*murder* documents), a weakly supervised approach is taken: the algorithm considers both in-domain and out-of-domain documents to learn what triplets are characteristic of the domain.

In the remainder of this section, we first look at preprocessing common to both methods, then describe each of the two methods in detail.

3.1. Common data preprocessing

Methods from Sections 3.2 and 3.3 operate on the same form of data representation, triplets. This section describes how they are constructed.

Starting with plain text, we first annotate it with some basic semantic and linguistic information. Using the ANNIE tool from the GATE framework [12], we first detect **named entities** and tag them as person, location or organization. We next use Enrycher [33] to perform **coreference and pronoun resolution** ("Mr. Obama", "President Barack Obama" and "he" might all refer to the same entity within an article). Finally, we use the Stanford parser [20] to obtain **dependency parses** for individual sentences. We simplify the parse trees using the following steps:

- For noun phrases, retain only the head of the phrase.
- Convert passive to active voice.
- Convert object-like relations (`dobj`, `acomp`, `infmod`, `nsubjpass`) to a simple "object" relation.
- Convert subject-like relations (`nsubj`, `agent`, `xsubj`) to a simple "subject" relation.
- Convert the `prep` relation to a "time", "location", or "instrument" relation or ignore it. The mapping is done based on ANNIE annotations and prepositional modifiers.
- Lemmatize all words.

We found the above simplifications to yield trees that are still sufficiently expressive but at the same time more semantic (e.g. normalizing away passive voice) and less complex, thus more likely to repeat across the corpus.

The simplified parse trees are used to derive **triplets**, the basic data representation structure for methods described in this paper. A triplet consists of two concepts and a relation connecting them; we experimented with two types of triplets:

- *verb–dependency–property*, which captures the verb and its main dependents. Using this representation, the sentence "*A violent tornado hit two houses in Texas*." produces $\boxed{\text{hit}} \xrightarrow{\text{subject}} \boxed{\text{tornado}}$, $\boxed{\text{hit}} \xrightarrow{\text{object}} \boxed{\text{house}}$ and $\boxed{\text{hit}} \xrightarrow{\text{location}} \boxed{\text{Texas}}$. The supported *dependency* values are subject, object, time, location and instrument. This representation is used by the Characteristic Triplet (CT) method in Section 3.3.
- *subject–verb–object*, which uses the verb itself as the relation. The example sentence above is expressed as $\boxed{\text{tornado}} \xrightarrow{\text{hit}} \boxed{\text{house}}$ with these triplets. This is more compact but captures only information encoded with transitive verbs. This representation is used by the Frequent Generalized Subgraph (FGS) method in Section 3.2 which uses redundant input data to alleviate the problem of capturing only transitive verbs.

As a last step, we **align** all triplets to a **knowledge base** (KB). We require a KB that a) is not domain-specific and b) is a simple ontology, in particular, covers the hypernymy relations between concepts. After evaluating Cyc [21] and WordNet [25], we decided for the latter because of its more pragmatic structure and better mappings to natural language. This is also supported by e.g. Boyd-Graber and Fellbaum [4] who note that "WordNet has become the lexical database of choice for NLP".

For each verb and property appearing in any of the triplets, we try to find the corresponding KB concept ("synset" in WordNet terminology). We first remove inflection from the words using python NLTK (Natural Language ToolKit), then align it to the corresponding synset. If more than one synset matches, we choose the most common sense; this is a proven approach and a very strong baseline for word sense disambiguation [23]. If no synset matches, we create a new one on the fly, expanding our local copy of WordNet. If the word for which the new concept was created (e.g. "Obama") was previously tagged by ANNIE as a person, location or organization, the new synset's hypernym is set accordingly. The new concepts are retained between algorithm runs.

As an alternative method of obtaining triplets, we also explored an approach based on Semantic Role Labeling (SRL) [34] which is a more natural fit for the task at hand than dependency parses. However, our conclusion was that the available language resources are unfortunately not yet mature enough to support open-domain tasks, which ours necessarily is. In particular, there is little training data beyond the few most frequent frames, and the linkage with other semantic resources is lacking.

The transformation of text to ontology-aligned triplets brings important **benefits**:

- *Feature selection*: only the key fragments of sentences are retained, following heuristics based on parse trees.
- *Noise reduction*: lower sensitivity to conjugation, tenses, synonyms, etc.
- *Access to background knowledge*: in our case, we exploit WordNet's hypernym taxonomy.

However, we also have to note several **limitations**:

- Both parsing and KB alignment introduce errors. Because the two steps are performed sequentially, their errors compound.
- The heuristic conversion of parse trees covers only specific (albeit the most common) types of expressions. For example, in the sentence "93 people were killed on Monday", 93 gets lost as it is only a modifier of the subject; and the sentence "President's visit to China ..." will not yield $\boxed{\text{President}} \xrightarrow{\text{visit}} \boxed{\text{China}}$ because "visit" here is not a verb.

We provide an analysis of the errors introduced by the semantic representation in Section 5.4.

Preprocessing real-world data. In principle, the methods can take any plain text as input. Evaluation, however, more realistically focuses on news data from the internet, a genre which is easily accessible. Section 4.1 describes our data acquisition and automatic cleartext extraction process. Together with the core methods in this section, they form a **full pipeline** leading from a few user-provided domain keywords to a domain schema for that domain.

3.2. FrequentGeneralized Subgraph (FGS)Method

This approach was first introduced in [34]. Here, we present additional details of the method and perform a quantitative evaluation which is missing in the original paper.

The key idea of the Frequent Generalized Subgraph (FGS) method is as follows: first, we construct a **semantic graph** for each document, consisting of triplet-derived entities and relations. Then, we mine graphs from all on-topic documents for **frequent subgraphs** whose specializations[1] appear in sufficiently many of those graphs. These generalized frequent subgraphs are what the method suggests as the topic template. The generalized nodes (e.g. $\boxed{person}$) and edges are the template slots and the graph as a whole provides context that makes it possible for humans to interpret the node.

In other words, the method assumes that while an individual triplet (e.g. $\boxed{person_1} \xrightarrow{\text{kill}} \boxed{person_2}$) may be frequent across multiple topics and its frequency

[1] "Specialization" in the sense of the hypernym taxonomy implied by our background knowledge base. For example, $\boxed{\text{Rodney}} \xleftarrow{\text{kill}} \boxed{\text{Wiley E.}} \xrightarrow{\text{detonate}} \boxed{\text{hand grenade}}$ is a specialization of $\boxed{person} \xleftarrow{\text{kill}} \boxed{person} \xrightarrow{\text{detonate}} \boxed{explosive}$.

does not attest to its suitability for a slot pattern, a small subgraph consisting of k or more nodes (e.g. for $k = 3$: $\boxed{person_2} \xleftarrow{\text{kill}} \boxed{person_1} \xrightarrow{\text{detonate}} \boxed{explosive}$) will only be frequent within a certain topic (here, suicide bomber attacks). k, the minimum "interesting" size of these subgraphs, is likely domain dependent. In our tests with news documents on five topics (see Section 4.1) we found $k = 3$ to give reasonable results.

Fig. 2 illustrates this with sample graphs from the "bombing attacks" domain. The graphs G_1, G_2 and G_3 each represent a semantic graph constructed from an input document. H is the generalized subgraph of all G_i and embodies a (partial) schema for the domain. In practice, the graphs G_i are larger, there are more of them and the subgraph H is only required to appear in some of the G_i.

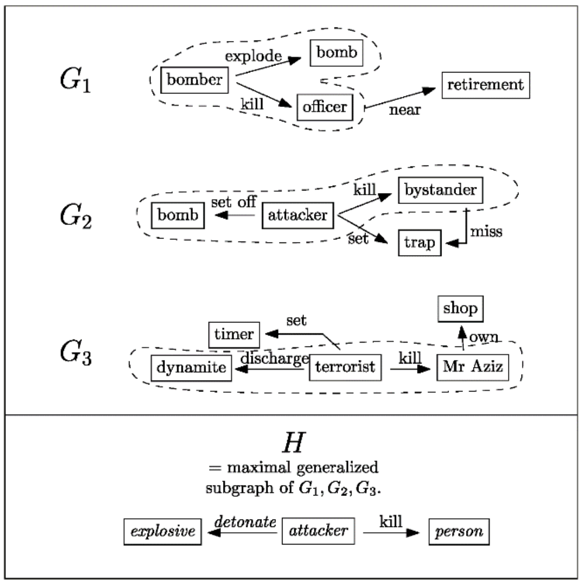


Figure 2. Example of a frequent generalized subgraph H as it would be identified by the FGS method for input graphs G_i. Each node in H has a specialization in G_i; e.g., "attacker" maps to "bomber" in G_1, "attacker" in G_2 and "terrorist" in G_3. This illustrative example is taken from the "bombing attack" domain and does not fully follow the WordNet hierarchy.

In the following subsections, we first briefly describe how the semantic graph is constructed, then turn to the technique for mining frequent subgraphs and to its generalization required by our approach.

3.2.1. Semantic Graph Construction

We construct the semantic graph from triplets derived in Section 3.1.We consider each triplet to be a 2-node graph, then treat the collection of all the triplets as a large disconnected graph and finally merge (collapse, identify) the nodes with the same labels.

The key simplifying assumption is that input documents tend to be focused in scope: we do not need to disambiguate entities other than by their labels. This is true of e.g. news articles, on which we evaluate. As an example, if an article mentions two buildings, one of which burns down and the second of which acted as a shelter for the fire fugitives, our method detects a single "building" and assigns both properties to it. Although having a means of distinguishing between the two would clearly be preferable, we have found this simplification not to cause significant issues in the newswire domain: entities which do need to be disambiguated are almost always presented with more unique names ("France" instead of "country" etc.). This rationale would have to be revised if one wanted to apply the approach to texts that are broader in focus.

Combating data sparsity. This method mines subgraphs that are frequent across individual article graphs. However, article graphs tend to be small, each capturing only a part of the information conveyed in the article due to the limited recall of the triplet representation. In addition, even when two documents convey the same information, they do not necessarily produce overlapping subgraphs (e.g. $\boxed{\text{Tom}} \xrightarrow{\text{drink}} \boxed{\text{glass}}$ vs. $\boxed{\text{Tom}} \xrightarrow{\text{have}} \boxed{\text{drink}}$). In fact, experiments show that for newswire, documents from the same domain almost never share subgraphs with three or more nodes.

This issue can be resolved with the use of parallel corpora. We thus derive each graph not from a single document but from the (textual) concatenation of multiple "parallel" documents that convey almost the same information, but paraphrased. As input, the FGS method therefore requires a document set that is further comprised of groups of parallel documents. Depending on the domain, this can be a serious limitation. However, for domains represented in the news, such data is available readily: we can exploit the fact that every noteworthy event is described in several news articles, and they form the required set of parallel documents.

We conduct our experiments on newswire data and derive each graph from the concatenation of 20–50 news articles from different sources that are all reporting on the same story. We observe this provides enough redundancy for subgraph patterns to occur across different *story* graphs. The number of articles per story that gives satisfactory results is likely dependent on the domain and uniformity of language (e.g. sports match reports are more formulaic than movie reviews); we tested with the above figure (20–50) as this is a common number of articles for real-world news stories, and did not experiment further. Details on how we acquire stories (i.e. clusters of articles) from a specific domain are given in Section 4.1.

3.2.2. Frequent Generalized Subgraph Mining

As described in Section 3.2, the method requires us to find frequent subgraph(s) of input graphs in a generalized manner, taking the hypernym taxonomy into account. This is non-trivial.

Formal problem statement. (Refer to Fig. 2 for easier understanding.) Given a set of labeled graphs $S = \{G_1, \dots, G_n\}$, a transitive antisymmetric relation on graph labels $genl(\cdot,\cdot)$ (with $genl(l', l)$ interpreted as "label l' is a generalization of label l") and a threshold $\theta \in \mathbb{N}$, we wish to construct all graphs H that are *generalized subgraphs* of at least θ graphs from S. A graph H is said to be a generalized subgraph of G if there is a mapping f of vertices $V(H)$ onto a subset of $V(G)$ such that $genl(v, f(v))$ holds for all $v \in V(H)$, and analogously for edges.

We are only interested in those H that are maximal in size, i.e. there is no graph $H^* \supsetneqq H$ such that H generalizes H^* and H^* also satisfies the above criteria. Among those, we only seek H that are as specific as possible.

This is computationally an exceptionally hard problem. Even finding frequent subgraphs verbatim–without taking possible generalizations (hypernyms) into account–presents a search space of subgraphs that grows exponentially with their size, and isomorphisms make even naive counting non-trivial. Extending the problem with generalizations as we do makes the search space even larger: each node in graphs $\{G_1, \dots, G_n\}$ can be independently generalized in multiple ways[2], making for yet another exponential growth factor.

We alleviate the generalization problem as follows: first, we transform all input graphs by generalizing each input node to the third level of the WordNet hierarchy and each input edge to its corresponding root in the WordNet hierarchy (edge labels are verbs, which do not have a common "*entity*"-like root). We found this to yield graphs that are general enough to generate desirable patterns and specific enough not to conflate unrelated patterns. Then, we perform regular frequent subgraph mining on these graphs to obtain candidates for subgraphs H as they are defined in the formal problem statement. The subgraphs obtained this way are typically overly generalized, so we specialize them back as much as possible without the support falling below θ. The respecialization is performed greedily: when multiple specializations are possible, which is almost always the case, we choose the one that has the highest support in input data.

Regular frequent subgraph mining in itself can be problematic. We had three modern dedicated programs (gSpan [37], Gaston [27] and HybridTreeMiner [10]) crash on our graphs with tens of thousands of nodes and thousands of labels (but work on smaller graphs), so we implemented our own solution based heavily on their ideas. The approach works in a way reminiscent of the classic a priori algorithm in frequent itemset mining: start with the smallest possible frequent graphs, i.e. those on one node, then iteratively add more and more nodes to them, discarding all graphs with an overly low support at each iteration. The algorithm is described in more detail in [34], and the implementation released at http://mitjat.com/research/.

[2] For example, possible generalizations of $\boxed{\text{suicide\_bomber}}$ are $\boxed{terrorist}$, $\boxed{radical}$, $\boxed{person}$ and $\boxed{entity}$

3.3. Characteristic Triplet (CT) Method

The Characteristic Triplet method is the second approach to constructing topic templates we propose. Its key idea is to find triplets which are frequent in documents belonging to the topic, yet infrequent in documents not belonging to it. Frequency is again considered in a generalized sense: $\boxed{\text{Obama}}$ contributes to the counts of $\boxed{politician}$, $\boxed{person}$ and $\boxed{entity}$. As with the FGS method, we are not searching for triplets that appear in the input documents verbatim but rather for their generalizations. For example, for the topic "political visits", we are looking for $\boxed{politician} \xrightarrow{\text{visit}} \boxed{country}$, which likely does not appear in any of the input documents.

The algorithm is based on the expectation that for any given topic, triplets (both the verbatim and generalized ones) will fit into one of the three categories below. Illustrative examples are given for the "diplomatic visits" domain:

- The **overly specific** triplets (e.g. $\boxed{\dots} \xrightarrow{\dots} \boxed{\text{Obama}}$) and the irrelevant ones (e.g. $\boxed{\dots} \xrightarrow{\dots} \boxed{football\ player}$) will have a low frequency count.
- The **overly generalized** triplets (e.g. $\boxed{\dots} \xrightarrow{\dots} \boxed{entity}$) will be frequent in on-topic documents but also off-topic ones.
- The triplets that are generalized "**just right**" (e.g. $\boxed{\dots} \xrightarrow{\dots} \boxed{politician}$) will be frequent in on-topic documents but less frequent otherwise; these are the ones we aim to detect.

The remainder of this section describes the algorithm based on this idea. We collect all triplets from input documents and all their generalizations and assign assign them scores that reflect the above intuition. The highest-scoring triplets form the topic template.

3.3.1. Triplet Lattice

The method assumes, in addition to the on-topic documents, a number of off-topic plain-text documents representative of the **background language**. This helps the method construct patterns that are general enough to be frequent only in the target domain, but specific enough to not be too frequent in the language in general. A similar idea has been successfully used in information extraction before: for example by Riloff [30] to judge the relevancy of extraction patterns, and by Yangarber [38] to stop the bootstrapping process of expanding the set of patterns at the right time.

We start by representing each document as a set of *verb–dependency–property* triplets as previously described in Section 3.1. Note that this is slightly different from the *subject–verb–object* triplets used in the FGS method; this alternative representation makes the method less susceptible to data sparsity in triplet space (see also "Combating data sparsity" in Section 03.2.1).

We next **construct a lattice** of triplets encountered in the input documents and their generalizations. Let us denote with c' the direct generalization (hypernym) of a concept c[3]. We initialize the lattice with every triplet $\boxed{v} \xrightarrow{d} \boxed{p}$ appearing verbatim in the input documents. Note that the points of the lattice are triplets which themselves are considered atomic. We then recursively extend the lattice by assigning to each triplet $\boxed{v} \xrightarrow{d} \boxed{p}$ as its parents the triplets $\boxed{v'} \xrightarrow{d} \boxed{p}$ and $\boxed{v} \xrightarrow{d} \boxed{p'}$, until reaching the root. See Fig. 3 for an illustration. Because the lattice is constructed using the hypernymy relation, it is a DAG (directed acyclic graph) and implies a partial order relation.

3.3.2. Cutting the Lattice

Each triplet t in the lattice is assigned a $frequency\ count$, defined as the number of times t or its specializations appear in on-topic documents. Formally, let $t \geq t^*$ denote that t is above t^* in the lattice, and let T_+ and T_0 denote the multiset of triplets in the on-topic documents and in the entire corpus, respectively. In T_+ and T_0, each triplet is counted once per source document. Then we define the frequency count of triplet t in on-topic documents as

$$f_+(t) \coloneqq |\{t^*: t^* \in T_+, t \geq t^*\}|.$$

Analoguously, we define $f_0(t)$ as the frequency count of t in the whole corpus. The value of $f_+(t)$ is also illustrated in Fig. 3; note how the off-topic documents do not contribute to $f_+(t)$ and how the value is not necessarily the sum of values in t's children.

Additionally, we assign a *score* to each triplet t in the lattice. The score $s(t)$ is tf-idf inspired:

$$s(t) \coloneqq f_+(t) \cdot log \frac{|T_0|}{f_0(t)}$$

Intuitively, the first factor favors more generalized triplets (as their frequency in on-topic documents is by definition higher than the frequency of highly specialized triplets) while the second factor assigns a lower score to triplets that overly generalized and appear frequently even in off-topic documents.

[3] A small fraction (1.7%) of entities in WordNet have multiple hypernyms. In this case, we use only the first hypernym, which is the more significant and, according to [17], by far the most relevant in 85% of the cases.

The scores $s(t)$ form the basis for selecting the triplets that will form the topic template. In Fig. 3, the triplet $\boxed{\text{destroy}} \xrightarrow{\text{obj}} \boxed{building}$ and its two parent triplets have the highest $f_+(\cdot)$. However, $\boxed{\text{destroy}} \xrightarrow{\text{obj}} \boxed{artifact}$ has a lower score than the other two since it also appears in the two non-topical documents.

3.3.3. Triplet Respecialization

We now take the 1000 top-scoring triplets as initial candidates for the final template. The cut-off of 1000 is conservative and was determined empirically; we observed that the relevant triplets typically have a much higher ranking. Next, we perform postprocessing to discard some of these candidate triplets for one of two reasons.

First, some frequent triplets are simply characteristic of the domain and always appear in the same form, e.g. $\boxed{\text{ground}} \xleftarrow{\text{subj}} \boxed{\text{shake}}$ for the domain of earthquake reports. Triplets that do not have multiple specializations in the input documents cannot possibly represent topic slots and are thus removed.

Second, based on our analysis of the test data, more than 90% of the high-scoring triplets are redundant and should be removed as well. As an example, consider Fig. 3 and assume *unmake* is the direct hypernym of *destroy* and *disassemble* in WordNet. Then, if even a single $\boxed{disassemble} \xrightarrow{\text{obj}} \boxed{\text{building}}$ were (possibly erroneously) detected in the on-topic documents, $\boxed{unmake} \xrightarrow{\text{obj}} \boxed{\text{building}}$ would have a higher score than $\boxed{destroy} \xrightarrow{\text{obj}} \boxed{\text{building}}$ even though $\boxed{unmake}$ "earned" most of its score through $\boxed{destroy}$. More generally, when we climb the lattice the score is *monotonically non-decreasing* as long as we don't encounter triplets that have specializations occurring in off-topic documents.

As a special case of the redundancy problem, observe that for any positively scored triplet t and its every generalization t' with no other descendants appearing in the input text, we have $s(t') = s(t)$. For example, in Fig. 3, there are no documents, either on– or off-topic, that contain a specialization of $\boxed{unmake} \xrightarrow{\text{obj}} \boxed{\text{building}}$ other than through $\boxed{destroy} \xrightarrow{\text{obj}} \boxed{\text{building}}$. Therefore, the two triplets have the same score even though the more specialized version is clearly preferable as output.

We correct for these effects by discarding all triplets t which have one or more children t∗ such that $s(t^*) > 0.80s(t)$. Here, 0.80 is a parameter that we fixed with a grid search. It is fairly robust; values in the range from 0.75 to 0.90 all gave comparable

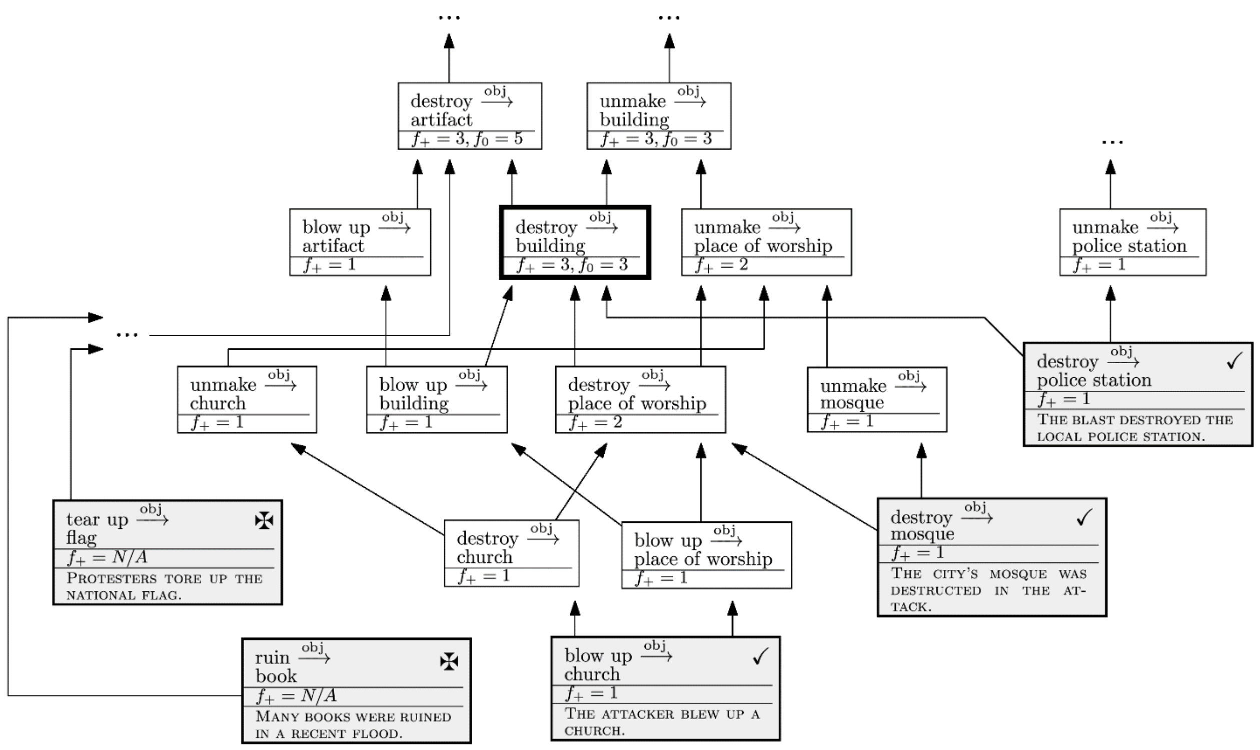

...
...
destroy $\xrightarrow{obj}$ artifact
$f_+ = 3, f_0 = 5$
unmake $\xrightarrow{obj}$ building
$f_+ = 3, f_0 = 3$
...
blow up $\xrightarrow{obj}$ artifact
$f_+ = 1$
destroy $\xrightarrow{obj}$ building
$f_+ = 3, f_0 = 3$
unmake $\xrightarrow{obj}$ place of worship
$f_+ = 2$
unmake $\xrightarrow{obj}$ police station
$f_+ = 1$
...
unmake $\xrightarrow{obj}$ church
$f_+ = 1$
blow up $\xrightarrow{obj}$ building
$f_+ = 1$
destroy $\xrightarrow{obj}$ place of worship
$f_+ = 2$
unmake $\xrightarrow{obj}$ mosque
$f_+ = 1$
destroy $\xrightarrow{obj}$ police station ✓
$f_+ = 1$
THE BLAST DESTROYED THE LOCAL POLICE STATION.
tear up $\xrightarrow{obj}$ flag ✠
$f_+ = N/A$
PROTESTERS TORE UP THE NATIONAL FLAG.
destroy $\xrightarrow{obj}$ church
$f_+ = 1$
blow up $\xrightarrow{obj}$ place of worship
$f_+ = 1$
destroy $\xrightarrow{obj}$ mosque ✓
$f_+ = 1$
THE CITY'S MOSQUE WAS DESTRUCTED IN THE ATTACK.
ruin $\xrightarrow{obj}$ book ✠
$f_+ = N/A$
MANY BOOKS WERE RUINED IN A RECENT FLOOD.
blow up $\xrightarrow{obj}$ church ✓
$f_+ = 1$
THE ATTACKER BLEW UP A CHURCH.

Figure 3. An example of a triplet lattice as constructed by the Characteristic Triplet (CT) method. Each box shows a triplet and its frequency f_+ in the on-topic documents. Here, the topic is "bombing attack". Each grey box represents a triplet that appears verbatim in an on-topic (X) or off-topic (m) input document. Grey boxes also contain the sentence that gives raise to the triplet. Arrows point from less generalized to more generalized triplets. The thick-bordered box represents the triplet with the highest score that gets selected for the template. The scores are related to the frequency f+ but not shown here; see Section 3.3 for discussion.

results. In Fig. 3, the triplet $\boxed{\text{unmake}} \xrightarrow{\text{obj}} \boxed{\textit{building}}$ is discarded in favor of $\boxed{\text{destroy}} \xrightarrow{\text{obj}} \boxed{\textit{building}}$ since it has the same score.

At the end of this pruning process, the remaining triplets are output as the template.

3.3.4. Frequent Generalized Subgraph (FGS) vs Characteristic Triplet (CT) Method

Note that like the FGS method described in the previous section, the CT method operates in the space of triplets. However, it makes several notable improvements:

- CT does not treat each topic in isolation but rather in relation to the background corpus distribution.
- By operating on structurally less complex units (triplets instead of subgraphs), CT does not require clusters of tightly related documents as input (see "Combating data sparsity" in Section 3.2.1).
- Due to not having to perform complex frequent subgraph mining, the CT method scales considerably better. Its time complexity is linear in the size of the input data.
- FGS expects a high level of regularity in the data to detect patterns, an expectation that often goes unfulfilled. CT is more flexible (and can therefore detect a higher number of patterns, as the evaluation later on also shows).

4. Evaluation

Evaluation of domain templates is not straightforward[4], to the point that many related articles only evaluate qualitatively (i.e. show a selected part of the output) or evaluate other aspects of their methods.

We evaluate on five newswire domains, comparing three methods: our FGS and CT, and a state of the art baseline. Section 4.1 describes the data and Section 4.2 details the methodology for evaluating this research problem.

We choose to adopt the evaluation method by Filatova [14] rather than the one by Chambers [7] and Qiu [9] based on MUC-4 data. As outlined in Section 2 (Related Work), we believe the latter to have several deficiencies. Most notably, it judges the quality of templates indirectly, through the lens of Information Extraction, while we evaluate the templates directly. It should still be noted that our evaluation only reflects the performance on a limited set of "real world" scenarios.

[4] A related article [28] notes, "While [template creation] is a difficult problem, its evaluation is arguably more difficult due to the dearth of suitable resources."

4.1. Datasets

We evaluated the algorithms on five domains/ topics, each captured by a set of news articles. We give the datasets single-word names:

- **airplane** - Reports of aircraft crashes.
- **bomb** - Reports of terrorist attacks (often by suicide bombers).
- **earthquake** - Reports of past earthquakes.
- **sentence** - Reports of sentencings passed in a court of law.
- **visit** - Reports of diplomatic visits by politicians.

We chose the topics based on what is covered by the media and based on the choices made by [14], the work we compare with. They evaluate on four domains: airplane crashes, terrorist attacks, earthquakes, and presidential elections. However, for the presidential elections domain they discover it is ill-defined and a poorly suited for evaluation.

Obtaining documents for a domain. We mentioned in the introduction that for domains that are reasonably well-represented in online news, we can obtain on-topic documents with minimal user input. Here, we briefly outline our process, although this is just one of the possible sources of common-topic document collections and not the focus of this work.

We assume the availability of a news collection, for example one obtained by crawling RSS feeds. We can then perform a simple query on the dataset with domain-related keywords–which are the only required user input. The results of such a naive search will be noisy: some matching documents will contain the query words by chance, unless the query is very elaborate; and at the same time, relevant documents may not match the exact query keywords. Our solution to this problem was to first create clusters of articles reporting on the same event, then retrieve *clusters* in which more than 25% percent of the articles match the query. This produced very clean results: even with no trial-and-error with the queries, more than 90% of the resulting articles were on the topic we had in mind when choosing the query.

There is ample existing work on clustering text streams and even news data streams in particular. The quality tends to be high because algorithms can heavily exploit the time component and effectively only cluster a day's or so worth of events. In our evaluation data, we used clusters as provided by Google News–a historical artifact of our past experiments. However, we later also found a reimplementation of [1] to give clusters that are indistinguishable in quality from Google's for the purpose of this data collection task.

Alternatively, one can obtain on-topic documents by directly querying a search engine and trust them to pick relevant documents; this was done for example by Chambers et al. [8]. This approach is even simpler and requires no offline data or clustering, but comes with obvious caveats regarding reproducibility and the amount of accessible data; we did not use it in our work.

Evaluation data. For the evaluation in this paper, we obtained on-topic documents using the method outlined above. Specifically, we queried a one-year crawl (August 2012 to July 2013) of Google News with the following expressions:

- **airplane**: (helicopter ∨ airplane) ∧ crash
- **bomb**: bomb ∧ attack
- **sentence**: judge ∧ court ∧ sentence
- **earthquake**: earthquake ∧ magnitude
- **visit**: (president ∨ minister ∨ diplomat) ∧ (meeting ∨ summit)

Those were the first queries we tried—we judge them to produce articles that are a good representtation of the topics as defined above. Tweaking the queries might produce more favorable end results in evaluation.

The full data crawl consists of 4 million articles clustered into 80 000 stories. The queries yielded more than 1000 results each; however, we only needed and used a random subset of 300 articles per topic for the evaluation.

In addition, we kept a random set of 4000 articles from the full dataset; those represent the background distribution (for the CT method) and are with relatively high probability not on-topic for any of our topics.

The downloaded articles were converted from HTML to plain text using the method presented in [35].

We provide our entire dataset online (see Appendix A).

4.2. Evaluation Methodology

What constitutes a good domain template? We characterize them as follows:

- A template should be **predictive of expected document content** within a domain. In other words, it should reflect the types of information humans expect to see in documents on that topic.
- A template should be **representative** of the domain, i.e. largely independent of the specific training data and not overfit to single aspects of it.

The first property, in particular, is hard to evaluate, and there is no established methodology. We are therefore devoting an entire section to proposing one.

To maximize reproducibility of results, we need to create a golden standard, i.e. the "ideal" template for every domain we wish to evaluate on. There are two problems associated with creating a golden standard:

- **Golden standards are noisy.** Like the better-known problem of summarization, our problem is inherently weakly defined; the notion of the "best" template differs from human to human. In our case, the problem is even more pronounced because it turns out people do not easily understand what a template/schema is.

- **Comparing with the golden standard.** Because of the expressivity of natural language, it is possible to obtain an output that is lexically and syntactically largely different from the golden standard, but semantically closely related[5]. This is again a problem faced when evaluating summaryzation algorithms.

4.2.1. Creating the Golden Standard

We combat the first problem listed above by disguising our task: we ask evaluators to have a look at some domain documents and then **pose 10 questions** that they believe would best help them summarize a **new, unseen document** from the domain if they got answers to them. This idea is largely due to Filatova [14].

We used the TaskRabbit[6] platform to recruit evaluators. The workers were not required to be domain experts, i.e. they had common-sense understanding of the domains only. They were native English speakers and were not in any way affiliated with the research. For use on potential new domains, we made available the exact phrasing (which proved to be very important and took refining) of the instructions given to workers; see Appendix A. We used three workers for each task.

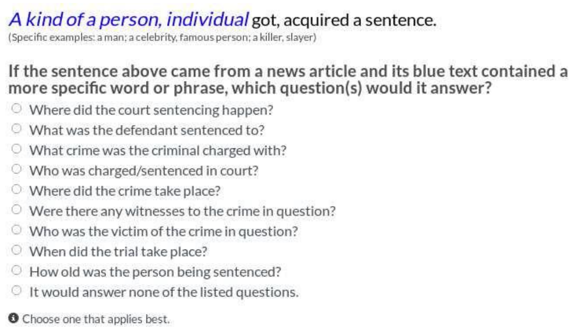
Finally, we revised and aggregated the questions ourselves. About a quarter of questions was discarded because they did not follow instructions. They tended to fall into two categories: 1) questions obviously referring to a single article instead of the topic in general and 2) metadata questions, e.g. "Who is reporting?", "Where was the article published?" etc. Within the remaining questions, we identified synonymous ones and retained the top 10 questions based on the number of times they were asked by our evaluators. Ties were broken by an unaffiliated friendly colleague in the hallway. These remaining *golden questions* form the golden standard. Table 2 lists the most popular questions for the "bombing attack" domain.

Table 2. Sample golden questions for the "bombing attack" domain

Sample golden questions
Who was killed?
Who was injured?
Which organization is suspected / admitted responsibility?
Where did the event happen?
Who was the bomb intended for?

[5] or example `X was shot` and `X took a hit`.

[6] http://taskrabbit.com; it differs from typical crowdsourcing platforms in that the tasks are larger and the involvement with workers more personal.


A kind of a person, individual got, acquired a sentence.
(Specific examples: a man; a celebrity, famous person; a killer, slayer)

If the sentence above came from a news article and its blue text contained a more specific word or phrase, which question(s) would it answer?

○ Where did the court sentencing happen?
○ What was the defendant sentenced to?
○ What crime was the criminal charged with?
○ Who was charged/sentenced in court?
○ Where did the crime take place?
○ Were there any witnesses to the crime in question?
○ Who was the victim of the crime in question?
○ When did the trial take place?
○ How old was the person being sentenced?
○ It would answer none of the listed questions.

Choose one that applies best.

Figure 4. A sample CrowdFlower task/unit. Evaluators judge if a triple (here $person \xrightarrow{\text{get}}$ sentence) corresponds to a golden question.

A golden standard in the form of natural-language questions has another advantage: it does not impose a representation or format on the algorithm output. This potentially allows a greater number of algorithms to be compared against each other, especially with the domain schema construction problem where the community has not yet converged on a single schema representation.

4.2.2. Comparing Against the Golden Standard

The downside of our golden standard is that we can not reliably automatically determine to what degree a proposed template matches it.

We therefore evaluate manually, using the CrowdFlower[7] (CF) crowdsourcing platform. We present the workers with a form that allows them to mark, for each output triplet, the golden question for which the triplet entails the answer. They can also mark that the triplet answers no questions. In CF terms, one such triplet-questions pair is called a *unit*. An example is provided in Fig. 4.

We use two mechanisms to ensure the output from CF is of high quality. First, we use their built-in mechanism of "gold units" (unrelated to our "golden standard"): we provide the expected worker responses to five clear-cut units, and workers that do not to get them right are excluded from further evaluation. Additionally, we filter out workers that have a CF-internal trustworthiness score below 0.80. Each unit is answered by five workers.

Finally, precision is computed as the percentage of output triplets that answer some golden question. Recall is computed as the percentage of golden questions answered by at least one output triplet.

4.2.3. Gauging Generalizability

As mentioned at the beginning of Section 4.2, we also wish to verify that the templates are not overfitted to the training corpus; this is of particular concern with our approach that qualifies template slots with detailed type information. A slot might

[7] http://crowdflower.com/; a reseller for Mechanical Turk and other, smaller crowdsourcing platforms

look reasonable at the outset, e.g. $\boxed{\text{earthquake}} \xrightarrow{\text{hit}} \boxed{capital}$ captures the location of an earthquake, but in reality earthquakes do not only hit capital cities and $\boxed{city}$ is preferred to $\boxed{capital}$.

As this property is not of central importance, we measure it automatically by proxy. For each topic, we take at most 80% of topical documents and use them to construct the topic template. For the remaining held-out set of documents, we verify how many of their triplets can be aligned to (i.e., are specializations of) the template triplets. We are careful to make the training-vs-test cut so that no news story is split between the two sets, ensuring that matches observed in the held-out set are due to *topic*-specific, not storyspecific pattern triplets. This metric does not generalize to other datasets, but as we only aim to compare our own methods, this simple approach suffices.

5. Results and Discussion

5.1. Template Quality

This subsection describes results pertaining to the evaluation described in Section 4.2.2, *Comparing against the golden standard*. We compare ourselves with FVM [14], a state of the art method that is representative of a large group of related methods. The method is summarized in Section 2, *Related work*.

As described, we evaluated on a set of five scenarios; we cannot guarantee performance on an arbitrary scenario or use case. However, the test scenarios were chosen in advance and by virtue of them being common topics of real-world news reports, so they are likely a reasonable approximate indicator of achievable performance for tasks that involve structuring events from news data.

The only metric reported in the FVM paper is recall (i.e. percentage of answered golden questions) at 20 “patterns”, which are comparable to our triplets (see Table 4 for examples of both). The metric makes good sense: the generated templates are primarily intended for humans and a useful algorithm should discover as many relevant template slots as possible. At the same time, reviewing 20 candidate slots seems like a reasonable burden for a knowledge worker. We evaluate according to the same metric and give results in Table 3.

It is clear from the table that FGS generates relatively poor templates relative to the other two algorithms. However, CT and FVM are roughly comparable. Both methods are consistently able to cover about a half of golden questions with the automatically generated templates, with our method achieving a higher absolute score in two of the domains.

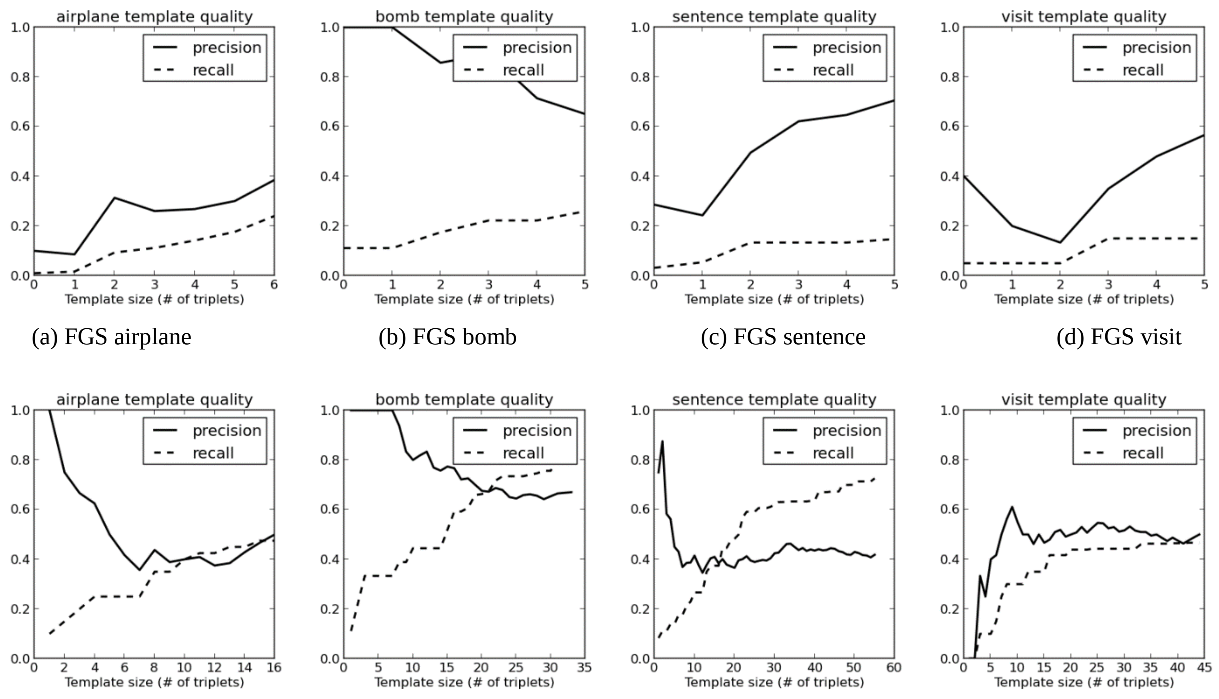
Table 3. Recall@20, i.e. the percentage of golden questions answered by top-20 template triplets. Comparison with state of the art (FVM) on their three domains and evaluation on two new ones.

Domain	FVM	FGS	CT
airplane	0.53	0.24	0.48
bomb	0.52	0.26	**0.66**
earthquake	0.38	—	**0.59**
visit	—	0.15	**0.48**
sentence	—	0.15	**0.44**

FVM authors did not evaluate on the `visit` and `sentence` domains. For the `earthquake` domain, the FGS method failed to discover any frequent subgraphs and thus produce a template. The cause seems to be a dispersed domain: articles report on earthquakes mostly in the context of related events (tsunamis, slides, rescue efforts, fundraisers etc.), so document graphs are quite different, share no large substructures and FGS fails.

We made a best-effort attempt to test in a setup comparable to that in the FVM paper; it was however not identical, and we therefore abstain from making strong claims about CT’s performance relative to FVM beyond observing that it appears at least comparable on the chosen domains. In particular, the FVM authors give the descriptions of the evaluated domains, but not the golden questions or the actual inputs documents, which were we reconstructed as detailed in Section 4. They extract templates from several hundred documents per topic; the exact number is not given, and we use 300 per topic. As a final detail, when preparing golden questions, the FVM authors do not merge individual worker’s questions into a single golden set and instead measure performance against each worker’s “golden” questions. The differences in measured performance across workers are however low, in the 5% range, so we use the average for the purpose of our comparison.

For our own methods, FGS and CT, we also provide precision and recall curves in Fig. 5. The figure further confirms that CT is preferred over FGS. Note also the different number of patterns extracted by each of the methods, i.e. the length of the x-axis: our assumptions when designing FGS were too strong; few subgraphs repeat across semantic graphs of different input articles, and only few patterns emerge. For CT, the number of extracted patterns varies. We were unable to find any quantitative property of the data for different domains which explains the variance. Qualitatively, the domains with more patterns simply have more varied reports. Differences in domains are to be expected; they are the primary reason to test on several domains in the first place.



(a) FGS airplane (b) FGS bomb (c) FGS sentence (d) FGS visit

(e) CT airplane (f) CT bomb (g) CT sentence (h) CT visit

Figure 5. Precision and recall of template triplets as measured by the golden standard

The irregular shapes of the precision curves show there is room for improvement in triplet ranking; whenever a high-quality topic triplet is ranked lower than a low-quality one, this causes an increase in the average precision and thus an upwards slope, while the precision curve of an ideally ranked set of template triplets would be monotonically decreasing. This discrepancy is particularly noticeable for the FGS method where a triplet “score” for the purposes of this plot is simply its frequency in input graphs, making for a poor ranking. The jagged lines are also the reason we chose an unorthodox but (in this case) more legible format for the precision-recall graphs. However, the overall precision is good, showing that our templates can facilitate manual domain schema construction.

Sample outputs. In Table 4, we show a **sample of patterns** produced by the three algorithms for the `bomb` domain. The *italic* text denotes template slots.

Note the highly detailed, automatically extracted slot types[8] in the output of our methods, which exploit background knowledge, compared to the output of FVM which operates on raw text and only abstracts away named entities (presumably with *number, date, person, location* and *organization*). Using a general purpose taxonomy like WordNet also allows us to identify slot fillers that are not named entities (hotel, mosque, policeman, ...), unlike the great majority of related work.

Reducing redundancy in the output set of triplets. Triplets as returned by existing methods are still not purely semantic: a fact can still be expressed with multiple triplets which are, as far as the ontology is concerned, unrelated (ex: $\boxed{\text{be\_after}} \xrightarrow{\text{obj}} \boxed{person}$ and $\boxed{\text{target}} \xrightarrow{\text{obj}} \boxed{personnel}$). We tried to make the results easier to interpret by clustering the pattern triplets post hoc. Two pattern triplets are considered more similar if their slots are more often filled with the same filler in the same story. Multiple similarity measures deriving from this intuition were tried, but none yielded satisfactory results, most likely due to data sparsity and the underconstrained nature of the problem. For example, $\boxed{\text{enter}} \xrightarrow{\text{obj}} \boxed{building}$ and $\boxed{\text{destroy}} \xrightarrow{\text{obj}} \boxed{building}$ were clustered by these methods because both triplets appear almost exclusively in articles related to bombing attacks, where they obviously strongly correlate. Given a much higher number of random non-bombing documents, the number of disconnected occurrences of $\boxed{\text{enter}} \xrightarrow{\text{obj}} \boxed{building}$ and $\boxed{\text{destroy}} \xrightarrow{\text{obj}} \boxed{building}$ would likely increase, possibly making the proposed approach effective. However, Yates et al. [39] report only 35% recall in identifying synonymous relations despite this being the primary goal of their paper; this proves that the problem is hard.

[8] Sometimes, statistics reveal more than we might expect – in determining that the location of a bombing attack is usually of type *Asian_country*, the CT method unknowingly makes a sad but true political commentary.

Table 4. Sample output from all three methods for the `bomb` domain. Template slots are shown in *italics*, **Ex** shows automatically extracted example values for the slot. All labels are taken directly from WordNet.

Frequent Verb Modifier (FVM)
killed (*number*) (NNS people)
(*person*) killed
(NN suicide) killed

Characteristic Triplet (CT)
$\text{kill} \xrightarrow{\text{object}} \mathit{defender/guardian}$ **Ex:** guard, constable, policeman
$\text{kill} \xrightarrow{\text{object}} \mathit{integer/whole\_number}$ **Ex:** 10, twelve, 15
$\text{target/aim} \xrightarrow{\text{object}} \mathit{force/personnel}$ **Ex:** police, military_personell
$\text{damage} \xrightarrow{\text{object}} \mathit{object\ vehicle}$ **Ex:** car, truck, airplane
$\text{destroy/destruct} \xrightarrow{\text{object}} \mathit{building/edifice}$ **Ex:** hotel, building, mosque
$\text{kill} \xrightarrow{\text{location}} \mathit{Asian\_country}$ **Ex:** Afghanistan, Pakistan, Iraq
$\text{kill} \xrightarrow{\text{object}} \mathit{city/metropolis}$ **Ex:** Beyrut, Kandahar, Bari
$\text{collar/nail (= arrest)} \xrightarrow{\text{time}} \mathit{weekday}$ **Ex:** Monday, Tuesday
$\mathit{attack/onslaught} \xleftarrow{\text{subject}} \text{come/come\_up}$ **Ex:** bombing, attack, foray/raid

Freq. Generalized Subgraph (FGS)
bomber – kill – *person/individual*   **Ex:** worshipper, policeman, civilian, person
bomb – kill – *integer/whole_number*   **Ex:** 10, one, two
person/individual – claim – duty/responsibility   **Ex:** leader, commandant
bomber – strike – *station*   **Ex:** police_station, terminal
person/individual – explode/detonate – explosive   **Ex:** man, soldier, militant

5.2. Alignment with Unseen Documents

This subsection gives the results of evaluation from Section 4.2.3, comparing the FGS in CT outputs in terms of how well they generalize to unseen data. The previous section shows that CT produces templates that human evaluators score as being more meaningful for their respective topics. However, this does not necessarily reveal much about how well the patterns represent the topics on the syntactic level. Do CT triplets also apply better to a held-out set of documents? Table 5 lists the AUC metric for a simple topical classifier: a document's score for domain d is defined as the sum of scores of template triplets that can be found in d. The classifier classifies into the domain with the highest score. Please note that we are not suggesting CT or FGS should be actually used for classification; their performance at this task is measured only to see which of the two produces a template that generalizes better to unseen data.

CT strongly outperforms FGS in this scenario as well. Analysis shows that this is mostly a direct consequence of the low number of patterns that FGS is able to suggest; its template is thus too restricted and relatively unlikely to fit unseen documents.

Although not relevant to relative comparison of CT's and FGS's performance, the variation in performance across domains is notable. The differences are expected; some domains are simply more dissimilar to the other domains and inherently easier to distinguish.

Table 5. Domain classification AUC in one-vs-all scenario.

Domain	**FGS**	**CT**
airplane	0.69	**0.83**
bomb	0.67	**0.71**
earthquake	0.50	**0.78**
sentence	0.73	**0.91**
visit	0.52	**0.82**

5.3. Robustness to Noise

Obtaining a dataset with *only* on-topic documents might not be easily feasible for a given domain. In particular, if the corpus is obtained using automated methods and/or heuristics, the data may be noisy. For the CT algorithm, we measured how noise impacts the performance of the algorithm.

Starting with the dataset used in the first experiment, we kept gradually adding random off-topic articles (i.e. noise) to the set of documents the algorithm believed to be on-topic, and measured how the recall@20 metric changes. The results are given in Fig. 6. The recall shown in the figure is averaged across topics; the variance was small.

We can observe a very high degree of robustness to noise. When the input collection of supposedly ontopic documents consists of 50% noise (which corresponds 300 documents in the graph), the impact on recall is still negligible. Only when the noisy documents significantly outnumber the true on-topic ones does performance decay. The reason for this behavior is that in the algorithm, the main contributor towards the score of a generalized triplet is its frequency. By adding noise, the frequency of the previously highscoring triplets does not decrease, whereas the triplets in the noisy part of the corpus do not have a coherent topic and are unlikely to produce

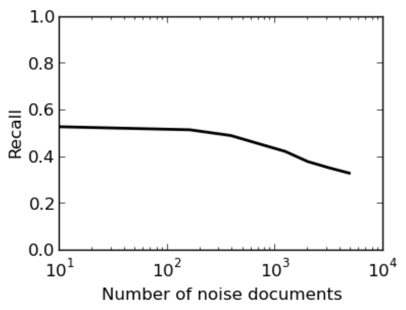


Figure 6. Robustness of the CT algorithm to noise. Graph shows the recall@20 as the original set of 300 on-topic documents is gradually tainted with additional random off-topic documents. The x axis is logarithmic.

a high-frequency pattern.

When enough noise is introduced, spurious high-frequency triplets do however get detected and suggested as patterns. They are typically highly generalized to accommodate the wide range of noisy inputs, for example $\boxed{person} \xleftarrow{\text{subj}} \boxed{perceive}$ (with *perceive* being a generalization of *feel*, *suffer*, *see*, *hear*, etc.).

5.4. Data Representation Error Analysis

Our data representation of choice—WordNet-aligned triplets—links the data to background knowledge and greatly simplifies its structure, making the two methods proposed in this paper feasible. However, to arrive at this representation, we make some strong assumptions. We next analyze how much error they introduce.

Triplet extraction. The triplet extraction process relies on an external dependency parser and a set of heuristics. We evaluated the quality of the extracted triplets from a representative text sample. We manually created a golden set of triplets for 50 sentences, each picked at random from a different (also random) online news article from our dataset. The sentences contain a total of 129 verbs involved in 210 triplets. For the task of correctly identifying triplet constituents (verbs, subjects, locations, ...) along with their dependency type, we measured an F_1 score of 78% (micro-averaged).

WordNet alignment. On the same set of 129 verbs and 210 modifiers, we also measured the performance of the "most common sense" word sense disambiguation heuristic. The measured accuracy was 76%. This is consistent with the 70–75% result reported in the literature [26, 19] for all-words word sense disambiguation with the same heuristic. (We only disambiguate noun and verb phrase headwords, which is likely somewhat easier.)

Note that even an incorrectly disambiguated word might still produce desired results. For verbs, the hypernym hierarchy is flat and the final patterns often contain verbs as they appeared in the text, without further generalizing them. When the pattern is finally presented to the user, it is semantically incorrect (as it is linked to the wrong WordNet concept) but looks correct, which might suffice depending on the use case. For nouns, the different senses of a word are sometimes very related and share the same hypernym: for example, `car.n.01` (an automobile) and `car.n.02` (a railway car) are both specializations of `wheeled_vehicle`. When a triplet involving the word *car* gets generalized during the template creation process, it does not matter any more whether it was initially disambiguated to the correct sense.

There are also cases where disambiguation goes critically wrong. For example, in the `bomb` domain, a relatively high-scoring pattern was $\boxed{\text{vehicle}} \xleftarrow{\text{subj}} \boxed{\text{kill}}$, largely the consequence of the word *bomber* being consistently incorrectly disambiguated as a bombing aircraft.

Sentence structure assumptions. One of the most limiting assumptions of the algorithm is that all extraction-worthy information is presented in sentences as a direct semantic dependents of the verb. Although this assumption is clearly too strong, it very usefully constrains the search space and has been adopted, possibly with minor variations, by the majority of related work. How much error does it introduce?

We reviewed the golden questions for which the CT algorithm was unable to find a corresponding pattern, and performed an informal analysis: for each question, we skimmed over several news articles to determine if it would be possible to answer the question with more expressive patterns.

1. **58%** of missed golden questions probably **could** be captured by relaxing the structural assumptions. The answers to these questions are mostly given as noun modifiers or as indirect objects with an appropriate modifier. Examples of missed questions: *(earthquake) What was the magnitude?* Typical constructs: "the 5.5-magnitude quake ...", "had a magnitude of 6.1". *(sentence) What crime was the criminal charged with?* Typical constucts: "sentenced for manslaughter", "guilty of knowingly participating in drug trafficking".
2. **42%** of missed golden questions probably **could not** be captured even with elaborate patterns. The corresponding information tends to be expressed in oblique ways that require either coreference resolution (a very hard problem in natural language processing) or even deeper reasoning/inference.
   Examples of missed questions*: (visit) How many countries were visited?* This would require the algorithm to note that the number of mentioned countries varies and is some-

how relevant. *(airplane) Who bears responsibility for the crash?* Many times, the information is not available; when it is, it is often given in a roundabout way: "a technical error was not to blame" (implying the pilot's fault), "an engine exploded" (implying it caused the crash).

This breakdown is largely anecdotal, but still gives a rough idea of the types of remaining challenges. We are particularly interested in expanding our triplet representation with preposition-bound indirect objects.

Illustrative error examples. Finally, in Table 4, we have intentionally included triplets that illustrate the limitations which any semantics-based (here, WordNet-based) approach likely has to face. First, the parsing of text into concepts and relations during preprocessing introduces errors that propagate through the pipeline. For example, "kill $\xrightarrow{\text{object}}$ *city/ metropolis*" from CT output is technically wrong – city is the location of the killing, not its object.

Second, the hypernym/hyponym distinctions in WordNet are sometimes very subtle, causing variation in content across documents to appear larger than it is. This causes, for example, the CT method to detect a slot *attack* with sample slot fillers *bombing*, *attack* and *raid*, between which people likely do not care to distinguish.

Third, while it certainly helps that WordNet collapses synonyms, sometimes the choice of the representative lemma for the synonym group (synset) is unusual or misleading. For example, the verb *collar/nail* in one of the CT triplets corresponds to the synset (WordNet concept) that also means "to arrest". Ideally, our algorithm should track the fact that this is the synset lemma that appeared in the text most often and use this lemma for display purposes.

6. Conclusions

In this work, we applied the "relational triplet" data representation to the task of unsupervised domain template construction for the first time, designing and implementing **two novel methods**. Both search the space of relational triplets to construct those that are not overly generic yet have strong support in the in-domain documents.

We evaluated on five domains and, using the CT method, achieved results that are at least comparable with current state of the art on a pre-selected set of domains in terms of quality while also providing **much finer-grained type information** about the template slots. While we cannot guarantee the usability of our method beyond the tested scenarios or in downstream use-cases, it seems likely that our method should do reasonably well at least on news data, since the test domains were chosen to be typical news events. The CT method has a linear time complexity and was shown to be highly robust to noise in the input data.

The presented methods are the first to approach the problem of domain template construction using **background knowledge** and explore its utility. We described the benefits and pitfalls of using the corresponding data representation and analyzed the ensuing errors.

Future work on this task and comparison of methods is facilitated by the detailed evaluation **methodology and datasets** with golden standards that we released.

Acknowledgments

This work was supported by the Slovenian Research Agency and the European Commission under PASCAL2 (IST-NoE-216886), ACTIVE (IST-2008-215040), RENDER (FP7-257790) and XLIKE (FP7-ICT-288342-STREP) and represents a part of doctoral studies at the University of Ljubljana. The authors would like to thank Primož Škraba for helpful discussions, Janez Brank for his thorough reading of the first draft, and the anonymous reviewers for their help with improving the article.

Appendix A - Datasets

All the data used for evaluation is available online at `http://mitjat.com/research/`. The package includes the following:

- Metadata (title, url, topic/domain) of all input documents used in our algorithms; see Section 4.1. 2000 nontopical and 5000 topical documents across 5 topics. Cleartext is available from the authors by request due to copyright concerns.
- A sample TaskRabbit instructions page (see Section 4.2.1; for reproducing the golden standard or expanding to new topics)
- The output of TaskRabbit workers and the actual golden questions used in our evaluation.
- A sample CrowdFlower task and instructions (see Section 4.2.2; for evaluating new algorithms)
- A `README.txt` describing the contents in more detail.

References

[1] **J. Azzopardi, C. Staff.** Incremental Clustering of News Reports. *Algorithms,* 2012, Vol. 5, No. 3, 364-378.

[2] **N. Balasubramanian, S. Soderland, O. E. Mausam, O. Etzioni.** Generating Coherent Event Schemas at Scale. In: *Proceedings of the Conference on Empirical Methods on Natural Language Processing EMNLP'13*, 2013, pp. 1721-1731.

[3] **M. Banko, O. Etzioni.** The tradeoffs between open and traditional relation extraction. In: *Proceedings of*

the Annual Meeting of the Association for Computational Linguistics ACL '08, Citeseer, 2008, pp. 28–36.

[4] **J. Boyd-Graber, C. Fellbaum.** Adding dense, weighted connections to WordNet. In: *Proceedings of the Third International WordNet Conference*, 2006, pp. 29–36.

[5] **A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. Hruschka Jr, T. Mitchell.** Toward an architecture for never-ending language learning. In: *Proceedings of the Twenty-Fourth Conference on Artificial Intelligence (AAAI 2010)*, 2010, pp. 1306–1313.

[6] **A. Carlson, J. Betteridge, R. Wang, E. Hruschka Jr, T. Mitchell.** Coupled semi-supervised learning for information extraction. In: *Proceedings of the third ACM international conference on Web search and data mining (WSDM), ACM*, 2010, pp. 101–110.

[7] **N. Chambers.** Event Schema Induction with a Probabilistic Entity-Driven Model. In: *Proceedings of the Conference on Empirical Methods on Natural Language Processing EMNLP '13*, 2013, pp. 1797–1807.

[8] **N. Chambers, D. Jurafsky.** Template-Based Information Extraction without the Templates. In: *Proceedings of the Annual Meeting of the Association for Computational Linguistics ACL'11*, 2011, pp. 976–986.

[9] **J. Cheung, H. Poon, L. Vanderwende.** Probabilistic frame induction. In: *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL/HLT 2013)*, 2013, pp. 837–846.

[10] **Y. Chi, Y. Yang, R. Muntz.** HybridTreeMiner: An efficient algorithm for mining frequent rooted trees and free trees using canonical forms. In: *Proceedings of 16th International Conference on Scientific and Statistical Database Management*, 2004, pp. 11–20.

[11] **D. Croce, C. Giannone, P. Annesi, R. Basili.** Towards Open-Domain Semantic Role Labeling. In: *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, July 2010*, pp. 237–246.

[12] **H. Cunningham, D. Maynard, K. Bontcheva.** Text Processing with GATE. *University of Sheffield Department of Computer Science*, 2011.

[13] **D. Das, M. Kumar, A. Rudnicky.** Automatic Extraction of Briefing Templates. In: *Proceedings of the International Joint Conference on Natural Language Processing IJCNLP '06*, 2008, pp. 265–272.

[14] **E. Filatova, V. Hatzivassiloglou, K. McKeown.** Automatic creation of domain templates. In: *Proceedings of the Annual Meeting of the Association for Computational Linguistics COLING/ACL '06, Morristown, NJ, USA, 2006, Association for Computational Linguistics*, pp. 207–214.

[15] **R. Grishman, B. Sundheim.** Message Understanding Conference-6: A Brief History. In: *Proceedings of the International Conference on Computational Linguistics COLING '96*, 1996, pp. 466–471.

[16] **A. Hotho, S. Staab, G. Stumme.** Text clustering based on background knowledge. *Tech. report, AIFB*, University of Karlsruhe, 2003.

[17] **F. Hristea.** Semiautomatic generation of wordnet type synsets and clusters using class methods–an overview. *Revue Roumaine de Linguistique*, 2007, Vol. 52, 97-133.

[18] **X. Kang, D. Li, S. Wang.** Research on domain ontology in different granulations based on concept lattice. *Knowledge-Based Systems*, 2012, Vol. 27, 152–161.

[19] **A. Kilgarriff.** How dominant is the commonest sense of a word?. *Text, Speech and Dialogue*, 2004, LNCS 3206, 103–111.

[20] **D. Klein, C. Manning.** Accurate unlexicalized parsing. In: *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics, Association for Computational Linguistics*, 2003, pp. 423–430.

[21] **D. Lenat.** CYC: A large-scale investment in knowledge infrastructure, *Communications of the ACM*, 1995, Vol. 38, 33-38.

[22] **Mausam, M. Schmitz, R. Bart, S. Soderland, O. Etzioni.** Open language learning for information extraction. In: *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, Association for Computational Linguistics*, July 2012, pp. 523–534.

[23] **D. McCarthy, R. Koeling, J. Weeds, J. Carroll.** Finding predominant word senses in untagged text. In: *Proceedings of the Annual Meeting of the Association for Computational Linguistics ACL'04*, 2004, pp. 280–287.

[24] **M. Michelson, C. Knoblock.** Constructing reference sets from unstructured, ungrammatical text. *Journal of Artificial Intelligence Research*, 2010, Vol. 38, 189-221.

[25] **G. Miller.** WordNet: a lexical database for English. *Communications of the ACM*, 1995, Vol. 38, 39-41.

[26] **R. Navigli.** Word sense disambiguation: A survey. *ACM Computing Surveys (CSUR)*, 2009, Vol. 41, 10:1-10:69.

[27] **S. Nijssen, J. Kok.** A quickstart in frequent structure mining can make a difference. In: *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, ACM*, 2004, pp. 647–652.

[28] **L. Qiu, M. Kan, T. Chua.** Modeling Context in Scenario Template Creation. In: *Proceedings of the Third International Joint Conference on Natural Language Processing IJCNLP '08*, 2008, pp. 157–164.

[29] **K. Radinsky, S. Davidovich.** Learning to predict from textual data. *Journal of Artificial Intelligence Research*, 2012, Vol. 45, 641-684.

[30] **E. Riloff.** Automatically generating extraction patterns from untagged text. In: *Proceedings of the Thirteenth National Conference on Artificial Intelligence AAAI '96*, 1996, pp. 1044–1049.

[31] **H. A. Santoso, S.-C. Haw, Z. Abdul-Mehdi.** Ontology extraction from relational database: Concept hierarchy as background knowledge. *Knowledge-Based Systems*, 2011, Vol. 24, 457-464.

[32] **Y. Shinyama, S. Sekine.** Preemptive information extraction using unrestricted relation discovery. In: *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics NAACL/HLT '06, Morristown, NJ, USA, June 2006, Association for Computational Linguistics*, pp. 304–311.

[33] **T. Stajner, D. Rusu, L. Dali, B. Fortuna, D. Mladenić, M. Grobelnik.** A service oriented

framework for natural language text enrichment. *Informatica (Ljubljana)*, 2010, Vol. 34, 307-313.

[34] **M. Trampuš, D. Mladenić.** Approximate Subgraph Matching for Detection of Topic Variations. In: *Proceedings of the 1st International Workshop on Knowledge Diversity on the Web (DiversiWeb 2011) at 20th International WWW Conference, Hyderabad, India*, 2011, pp. 25–28.

[35] **M. Trampuš, B. Novak.** Internals of an aggregated web news feed. In: *Proceedings of the Fifteenth International Information Science Conference IS SiKDD 2012*, 2012, pp. 431–434.

[36] **D. Vrandečić.** Wikidata: A new platform for collaborative data collection. In: *Proceedings of the 21st International Conference Companion on World Wide Web WWW'12*, 2012, pp. 1063–1064.

[37] **X. Yan, J. Han.** gSpan: Graph-based substructure pattern mining - UIUC Technical Report. Tech. report, University of Illinois at Urbana-Champaign, 2002.

[38] **R. Yangarber.** Counter-training in discovery of semantic patterns. In: *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-ACL'03*, Vol. 1, Morristown, NJ, USA, July 2003, Association for Computational Linguistics, pp. 343–350.

[39] **A. Yates, O. Etzioni.** Unsupervised methods for determining object and relation synonyms on the web. *Journal of Artificial Intelligence Research*, 2009, Vol. 34, 255-296.

[40] **S. Zelikovitz, H. Hirsh.** Integrating background knowledge into nearest-Neighbor text classification. *Advances in Case-Based Reasoning,* 2002, LNCS 2416, pp. 1–5.

Received April 2013.