Modeling Context in Scenario Template Creation

Long Qiu, Min-Yen Kan, Tat-Seng Chua
Department of Computer Science
National University of Singapore
Singapore, 117590
{qiul,kanmy,chuats}@comp.nus.edu.sg

Abstract

We describe a graph-based approach to Scenario Template Creation, which is the task of creating a representation of multiple related events, such as reports of different hurricane incidents. We argue that context is valuable to identify important, semantically similar text spans from which template slots could be generalized. To leverage context, we represent the input as a set of graphs where predicate-argument tuples are vertices and their contextual relations are edges. A context-sensitive clustering framework is then applied to obtain meaningful tuple clusters by examining their intrinsic and extrinsic similarities. The clustering framework uses Expectation Maximization to guide the clustering process. Experiments show that: 1) our approach generates high quality clusters, and 2) information extracted from the clusters is adequate to build high coverage templates.

1 Introduction

Scenario template creation (STC) is the problem of generating a common semantic representation from a set of input articles. For example, given multiple newswire articles on different hurricane incidents, an STC algorithm creates a template that may include slots for the storm's name, current location, direction of travel and magnitude. Slots in such a scenario template are often to be filled by salient entities in the scenario instance (e.g., "Hurricane Charley" or "the coast area") but some can also be filled by prominent clauses, verbs or adjectives that describe these salient entities. Here, we use the term *salient aspect* (SA) to refer to any of such slot fillers that people would regard as important to describe a particular scenario. Figure 1 shows such a manually-built scenario template in which details about important actions, actors, time and locations are coded as slots.

STC is an important task that has tangible benefits for many downstream applications. In the Message Understanding Conference (MUC), manually-generated STs were provided to guide Information Extraction (IE). An ST can also be viewed as regularizing a set of similar articles as a set of attribute/value tuples, enabling multi-document summarization from filled templates.

Despite these benefits, STC has not received much attention by the community. We believe this is because it is considered a difficult task that requires deep NL understanding of the source articles. A problem in applications requiring semantic similarity is that the same word in different contexts may have different senses and play different roles. Conversely, different words in similar contexts may play similar roles. This problem makes approaches that rely on word similarity alone inadequate.

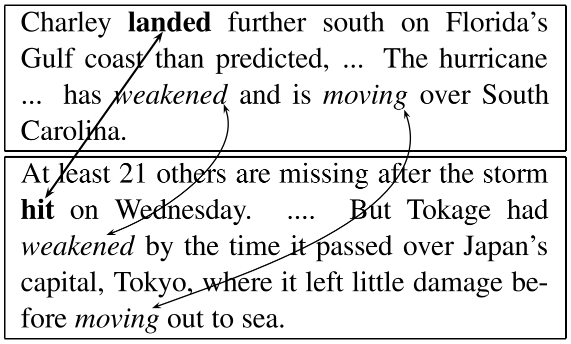
We propose a new approach to STC that incorporates the use of contextual information to address this challenge. Unlike previous approaches that concentrate on the intrinsic similarity of candidate slot fillers, our approach explicitly models contextual evidence. And unlike approaches to word sense disambiguation (WSD) and other semantic analyses that use neighboring or syntactically related words as contextual evidence, we define contexts by semantic relatedness which extends beyond sentence boundaries. Figure 2 illustrates a case in point with two excerpts from severe storm reports. Here, although the intrinsic similarity of the main verbs "hit" and "land" is low, their contextual similarity is high as both are followed by clauses sharing similar subjects (hurricanes) and the same verbs. Our approach encodes such contextual information as graphs, mapping the STC problem into a general graph overlay problem that is solvable by a variant of Expectation Maximization (EM).

Our work also contributes resources for STC research. Until now, few scenario templates have been publicly available (as part of MUC), rendering any potential evaluation of automated STC statistically insignificant. As part of our study, we have compiled a set of input articles with annotations that we are making available to the research community.

Scenario Template: Storm

Storm Name	*Charley*
Storm Action	*landed*
Location	*Florida's Gulf coast*
Time	*Friday at 1950GMT*
Speed	*145 mph*
Victim Category 1	*13 people*
Action	*died*
Victim Category 2	*over one million*
Action	*affected*

Figure 1: An example scenario template (filled).



Charley **landed** further south on Florida's Gulf coast than predicted, ... The hurricane ... has *weakened* and is *moving* over South Carolina.

At least 21 others are missing after the storm **hit** on Wednesday. .... But Tokage had *weakened* by the time it passed over Japan's capital, Tokyo, where it left little damage before *moving* out to sea.

Figure 2: Contextual evidence of similarity. Curved lines indicate similar contexts, providing evidence that "land" and "hit" from two articles are semantically similar.

2 Related Work

A natural way to automate the process of STC is to cluster similar text spans in the input article set. SAs then emerge through clustering; if a cluster of text spans is large enough, the aspects contained in it will be considered as SAs. Subsequently, these SAs will be generalized into one or more slots in the template, depending on the definition of the text span. Assuming scenarios are mainly defined by actions, the focus should be on finding appropriate clusters for text spans each of which represents an action. Most of the related work (although they may not directly address STC) shares this assumption and performs action clustering accordingly. While the target application varies, most systems that need to group text spans by similarity measures are verb-centric.

In addition to the verb, many systems expand their representation by including named entity tags (Collier, 1998; Yangarber et al., 2000; Sudo et al., 2003; Filatova et al., 2006), as well as restricting matches (using constraints on subtrees (Sudo et al., 2003; Filatova et al., 2006), predicate argument structures (Collier, 1998; Riloff and Schmelzenbach, 1998; Yangarber et al., 2000; Harabagiu and Maiorano, 2002) or semantic roles).

Given these representations, systems then cluster similar text spans. To our knowledge, all current systems use a binary notion of similarity, in which pairs of spans are either similar or not. How they determine similarity is tightly coupled with their text span representation. One criterion used is pattern overlap: for example, (Collier, 1998; Harabagiu and Lacatusu, 2005) judge text spans to be similar if they have similar verbs and share the same verb arguments. Working with tree structures, Sudo et al. and Filatova et al. instead require shared subtrees.

Calculating text span similarity ultimately boils down to calculating word phrase similarity. Approaches such as Yangarber's or Riloff and Schmelzenbach's do not employ a thesaurus and thus are easier to implement, but can suffer from over- or under-generalization. In certain cases, either the same actor is involved in different actions or different verbs realize the same action. Other systems (Collier, 1998; Sudo et al., 2003) do employ lexical similarity but threshold it to obtain binary judgments. Systems then rank clusters by cluster size and correlation with the relevant article set and equate top clusters as output scenario slots.

3 Context-Sensitive Clustering (CSC)

Automating STC requires handling a larger degree of variations than most previous work we have surveyed. Note that the actors involved in actions in a scenario generally differ from event to event, which makes most related work on text span similarity calculation unsuitable. Also, action participants are not limited to named entities, so our approach needs to process all NPs. As both actions and actors may be realized using different words, a similarity thesaurus is necessary. Our approach to STC uses a thesaurus based on corpus statistics (Lin, 1998) for real-valued similarity calculation. In contrast to previous approaches, we do not threshold word similarity results; we retain their fractional values and incorporate these values holistically. Finally, as the same action can be realized in different constructions, the semantic (not just syntactic) roles of verb arguments must be considered, lest agent and patient roles be confused. For these reasons, we use a semantic role labeler (Pradhan et al., 2004) to provide and delimit the text spans that contain the semantic arguments of a predicate. We term the obtained text spans as *predicate argument tuples* (tuples) throughout the paper. The semantic role labeler reportedly achieves an F_1 measure equal to 68.7% on identification-classification of predicates and core arguments on a newswire text corpus (LDC, 2002). Within the confines of our study, we find it is able to capture most of the tuples of interest.

Our approach explicitly captures contextual evidence. We define a tuple's contexts as other tuples in the same article segment where no topic shift occurs. This definition refines the n-surrounding word constraint commonly used in spelling correction (for example, (Hirst and Budanitsky, 2005)), Word Sense Disambiguation ((Preiss, 2001), (Lee and Ng, 2002), for instance), *etc.* while still ensures the relatedness between a tuple and its contexts. Specifically, a tuple is contextually related to other tuples by two quantifiable contextual relations: *argument-similarity* and *position-similarity*. For our experiments, we use the leads of newswire articles as they normally summarize the news. We also assume a lead qualifies as a single article segment, thus making all of its tuples as potential contexts to each other.

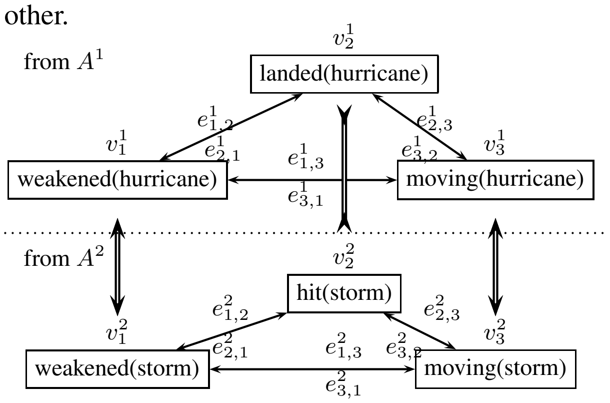


Figure 3: Being similar contexts, "weakened" and "moving" provide contextual evidence that "land" and "hit" are similar.

First, we split the input article leads into sentences and perform semantic role labeling immediately afterwards. Our system could potentially benefit from additional pre-processing such as co-reference resolution. Currently these pre-processing steps have not been properly integrated with the rest of the system, and thus we have not yet measured their impact.

We then transform each lead A^i into a graph $G^i = \{V^i, E^i\}$. As shown in Figure 3, vertices $V^i = \{v^i_j\}(j = 1, ..., N)$ are the N predicate argument tuples extracted from the ith article, and directed edges $E^i = \{e^i_{m,n} = (v^i_m, v^i_n)\}$ reflect contextual relations between tuple v^i_m and v^i_n. Edges only connect tuples from the same article, i.e., within each graph G^i. We differentiate between two types of edges. One is *argument-similarity*, where the two tuples have semantically similar arguments. This models tuple cohesiveness, where the edge weight is determined by the similarity score of the most similar inter-tuple argument pair. The other is *position-similarity*, represented as the offset of the ending tuple with respect to the other, measured in sentences. This edge type is directional to account for simple causality.

Given this set of graphs, the clustering task is to find an optimal alignment of all graphs (i.e., superimposing the set of article graphs to maximize vertex overlap, constrained by the edges). We adapt Expectation Maximization (Dempster et al., 1977) to find an optimal clustering. This process assigns tuples to suitable clusters where they are semantically similar and share similar contexts with other tuples. Algorithm 1 outlines this alignment process.

Algorithm 1 Graph Alignment($\mathcal{G}$)

```
/*$\mathcal{G}$ is a set of graph $\{G^i\}$*/
$T \leftarrow$ all tuples in $\mathcal{G}$
$C \leftarrow$ highly cohesive tuples clusters
$other \leftarrow$ remaining tuples semantically connected with $C$
$C[C.length] \leftarrow other$
repeat
    /*E step*/
    for each $i$ such that $i < C.length$ do
        for each $j$ such that $j < C.length$ do
            if $i == j$ then
                continue;
            re-estimate $parameters[C[i], C[j]]$ /*distribution
            parameters of edges between two clusters*/
    $tupleReassigned = false$ /*reset*/
    /*M step*/
    for each $i$ such that $i < T.length$ do
        $aBestLikelihood = T[i].likelihood$; /*likelihood of
        being in its current cluster*/
        for each tuple $t_{contxt}$ that contextually related with
        $T[i]$ do
            for each cluster $c_{cand}$, any candidate cluster that
            contextually related with $t_{contxt}.cluster$ do
                $P(T[i] \in c_{cand}) = comb(P_s, P_c)$
                $likelihood = log(P(T[i] \in c_{cand}))$
                if $likelihood > aBestLikelihood$ then
                    $aBestLikelihood = likelihood$
                    $T[i].cluster = c_{cand}$
                    $tupleReassigned = true$
until $tupleReassigned == false$ /*alignment stable*/
return
```

During initialization, tuples whose pairwise similarity higher than a threshold τ are merged to form highly cohesive seed clusters. To compute a continuous similarity $Sim(t_a, t_b)$ of tuples t_a and t_b, we use the similarity measure described in (Qiu et al., 2006), which linearly combines similarities between the semantic roles shared by the two tuples. Some other tuples are related to these seed clusters by *argument-similarity*. These related tuples are temporarily put into a special "other" cluster. The cluster membership of these related tuples, together with those currently in the seed clusters, are to be further adjusted. The "other" cluster is so called because a tuple will end up being assigned to it if it is not found to be similar to any other tuple. Tuples that are neither similar to nor contextually related by *argument-similarity* to another tuple are termed *singletons* and excluded from being clustered.

We then iteratively (re-)estimate clusters of tuples across the set of article graphs $\mathcal{G}$. In the E-step of the EM algorithm, all contextual relations between each pair of clusters are collected as two set of *edges*. Here we assume *argument-similarity* and *position-similarity* are independent and thus we differentiate them in the computation. Accordingly, there are two sets: $edges_{as}$ and $edges_{ps}$. For simplicity, we assume independent normal distributions for the strength of each set (inter-tuple argument similarity for $edges_{as}$ and sentence distance for $edges_{ps}$). The edge strength distribution parameters for both sets between each pair of clusters are re-estimated based on current edges in $edges_{as}$ and $edges_{ps}$.

In the M-step, we examine each tuple's fitness for belonging to its cluster and relocate some tuples to new clusters to maximize the likelihood given the latest estimated edge strength distributions. In the following equations, we denote the proposition that *predicate argument tuple t_a belongs to cluster c_m* as $t_a \in c_m$; a typical tuple (the centroid) of the cluster c_m as t_{c_m}; and the cluster of t_a as c_{t_a}. The objective function to maximize is:

$$Obj(\mathcal{G}) = \sum_{t_a \in \mathcal{G}} log(P(t_a \in c_{t_a})), \tag{1}$$

$$\text{where } P(t_a \in c_m) = \frac{2P_s(t_a \in c_m)\, P_c(t_a \in c_m)}{P_s(t_a \in c_m) + P_c(t_a \in c_m)}. \tag{2}$$

Equation 2 takes the harmonic mean of two factors: a contextual factor P_c and and a semantic factor P_s:

$$P_c(t_a \in c_m) = \max_{t_b: edges(t_a, t_b) \neq null}\{P(edges(t_a, t_b) \mid edges(c_m, c_{t_b}))\}, \tag{3}$$

$$P_s(t_a \in c_m) = \begin{cases} sim_{default}, & c_m = c_{other}, \\ Sim(t_a, t_{c_m}), & \text{otherwise.} \end{cases} \tag{4}$$

Here the contextual factor P_c models how likely t_a belongs to c_m according to the contextual information, i.e., the conditional probability of the contextual relations between c_m and c_{t_b} given the contextual relations between t_a and one particular context t_b, which maximizes this probability. According to Bayes' theorem, it is computed as shown in Equation 3. In practice, we multiply two conditional probabilities: $P(edge_{as}(t_a, t_b) | edges_{as}(c_m, c_{t_b}))$ and $P(edge_{ps}(t_a, t_b) | edges_{ps}(c_m, c_{t_b}))$, assuming independence between $edges_{as}$ and $edges_{ps}$.

We assume there are still singleton tuples that are not semantically similar to another tuple and should belong to the special "other" cluster. Given that they are dissimilar to each other, we set $sim_{default}$ to a small nonzero value in Equation 4 to prevent the "other" cluster from expelling them based on their low semantic similarity. Tuples' cluster memberships are recalculated, and the parameters describing the contextual relations between clusters are re-estimated. New EM iterations are performed as long as one or more tuple relocations occur. Once the EM halts, clusters of equivalent tuples are formed. Among these clusters, some correspond to salient actions that, together with their actors, are all SAs to be generalized into template slots. Cluster size is a good indicator of salience, and each large cluster (excluding the "other" cluster) can be viewed as containing instances of a salient action.

Formulating the clustering process as a variant of iterative EM is well-motivated as we consider the similarity scores as noisy and having missing observations. Calculating semantic similarity is at best inaccurate. Thus it is difficult to cluster tuples correctly based only on their semantic similarity. Also to check whether a tuple shares contexts with a cluster of tuples, the cluster has to be relatively clean. An iterative EM as we have proposed naturally improve the cleanness of these tuple clusters gradually as new similarity information comes to light.

4 Evaluation

For STC, we argue that it is crucial to cluster tuples with high recall so that an SA's various surface forms can be captured and the size of clusters can serve as a salience indicator. Meanwhile, precision should not be sacrificed, as more noise will hamper the downstream generalization process which outputs template slots. We conduct experiments designed to answer two relevant research questions:
1) **Cluster Quality:** Whether using contexts (in CSC) produces better clustering results than ignoring it (in the K-means baseline); and
2) **Template Coverage:** Whether slots generalized from CSC clusters cover human-defined templates.

4.1 Data Set and Baseline

A straightforward evaluation of a STC system would compare its output against manually-prepared gold standard templates, such as those found in MUC. Unfortunately, such scenario templates are severely limited and do not provide enough instances for a proper evaluation. To overcome this problem, we have prepared a balanced news corpus, where we have manually selected articles covering 15 scenarios. Each scenario is represented by a total of 45 to 50 articles which describe 10 different events.

Our baseline is a standard K-means clusterer. Its input is identical to that of CSC – the tuples extracted from relevant news articles and are not excluded from being clustered by CSC in the initialization stage (refer to Section 3) – and employs the same tuple similarity measure (Qiu et al., 2006). The differentiating factor between CSC and K-means is the use of contextual evidence. A standard K-means clusterer requires a k to be specified. For each scenario, we set its k as the number of clusters generated by CSC for direct comparison.

We fix the test set for each scenario as ten randomly selected news articles, each reporting a different instance of the scenario; the development set (which also serves as the training set for determining the EM initialization threshold τ and $sim_{default}$ in Equation 4) is a set of ten articles from the "AirlinerCrash" scenario, which are excluded from the test set. Both systems analyze the first 15 sentences of each article, and sentences generate 2 to 3 predicate argument tuples on average, resulting in a total of $10 \times 15 \times$ (2 to 3) = 300 to 450 tuples for each scenario.

4.2 Cluster Quality

This experiment compares the clustering results of CSC and K-means. We use the standard clustering metrics of *purity* and *inverse purity* (Hotho et al., 2003). The first author manually constructed the gold standard clusters for each scenario using a GUI before conducting any experiments. A special cluster, corresponding to the "other" cluster in the CSC clusters, was created to hold the singleton tuples for each scenario. Table 1 shows this under the column "#Gold Standard Clusters".

Using the manual clusters as the gold standard, we obtain the purity (P) and inverse purity (IP) scores of CSC and K-means on each scenario. In Table 1, we see that CSC outperforms K-means on 10 of 15 scenarios for both P and IP. For the remaining 5 scenarios, where CSC and K-means have comparable

P scores, the IP scores of CSC are all significantly higher than that of K-means. This suggests clusters tend to be split apart more in K-means than in CSC when they have similar purity. One thing worth mentioning here is that the “other” cluster normally is relatively large for each scenario, and thus may skew the results. To remove this effect, we excluded tuples belonging to the CSC “other” cluster from the K-means input, generating one fewer cluster. Running the evaluation again, the resulting P-IP scores again show that CSC outperforms the baseline K-means. We only report the results for all tuples in our paper for simplicity.

Scenario	#Gold Std. Clusters	CSC		K-means	
		P	IP	P	IP
AirlinerCrash	23	**.61**	**.42**	.52	.28
Earthquake	18	**.60**	**.44**	.53	.30
Election	10	**.77**	**.49**	.75	.21
Fire	14	**.65**	**.44**	.64	.26
LaunchEvent	12	**.77**	**.37**	.73	.22
Layoff	10	**.71**	**.28**	.70	.19
LegalCase	8	.75	**.37**	.75	.18
Nobel	6	.77	**.28**	.77	.19
Obituary	7	**.85**	**.46**	.81	.28
RoadAccident	20	**.61**	**.49**	.56	.40
SoccerFinal	5	.88	**.39**	.88	.15
Storm	14	.61	**.31**	.61	.22
Tennis	6	.87	**.19**	.87	.12
TerroristAttack	14	**.64**	**.48**	.62	.25
Volcano	16	**.68**	**.38**	.66	.17
Average	12.2	**.72**	**.39**	.69	.23

Table 1: CSC outperforms K-means with respect to the purity (P) and inverse purity (IP) scores.

A close inspection of the results reveals some problematic cases. One issue worth mentioning is that for certain actions both CSC and K-means produce split clusters. In the CSC case, we traced this problem back to the thesaurus, where predicates for one action seem to belong to two or more totally dissimilar semantic categories. The corresponding tuples are thus assigned to different clusters as their low semantic similarity forces the tuples to remain separate, despite the shared contexts trying to join them. One example is “blast (off)” and “lift (off)” in the “Launch Event” scenario. The thesaurus shows the two verbs are dissimilar and the corresponding tuples end up being in two split clusters. This can not be solved easily without an improved thesaurus. We are considering adding a prior to model the optimal size for clusters, which may help to compact such cases.

4.3 Template Coverage

We also assess how well the resulting, CSC-generated tuple clusters serve in creating good scenario template slots. We start from the top largest clusters from each scenario, and decompose each of them into six sets: the *predicates*, *agents*, *patients*, *predicate modifiers*, *agent modifiers* and *patient modifiers*. For each of the first three sets for each cluster, we create a generalized term to represent it using an extended version of a generalization algorithm (Tseng et al., 2006). These terms are deemed output slots, and are put into the template with their agent-predicate-patient relations preserved. The size of the template may increase when more clusters are generalized, as new slots may result.

We manually compare the slots that are output from the system with those defined in existing scenario templates in MUC. The results here are only indicative and not conclusive, as there are only two MUC7 templates available for comparison: *Aviation Disaster* and *Launch Event*.

Template	semantic role	general term
cluster 1	**action**	crash
	agent	aircraft
	patient	—
cluster 2	**action**	kill
	agent	heavier-than-air-craft
	patient	people

Figure 4: Automated scenario template of “AviationDisaster”.

Figure 4 shows an excerpt of the automatically generated template “AviationDisaster” (“Airliner-Crash” in our corpus) where the semantic roles in the top two biggest clusters have been generalized. Their modifiers are quite semantically diverse, as shown in Table 2. Thus, generalization (probably after a categorization operation) remains as a challenging problem.

Nonetheless, the information contained in these semantic roles and their modifiers covers human-

semantic role	modifier head samples
agent:aircraft	A, U.N., The, Swiss, Canadian-built, AN, China, CRJ-200, military, Iranian, Air, refueling, US, ...
action:crash	Siberia, mountain, rain, Tuesday, flight, Sharjah, flames, Sunday, board, Saturday, 225, Rockaway, approach, United, mountain, hillside
patient:people	all, 255, 71

Table 2: Sample automatically detected modifier heads of different semantic roles.

	AviationDisaster
*	AIRCRAFT
*	AIRLINE
	DEPARTURE_POINT
	DEPARTURE_DATE
*	AIRCRAFT_TYPE
*	CRASH_DATE
*	CRASH_SITE
	CAUSE_INFO
*	VICTIMS_NUM

	LaunchEvent
*	VEHICLE
*	VEHICLE_TYPE
*	VEHICLE_OWNER
*	PAYLOAD
	PAYLOAD_TYPE
	PAYLOAD_FUNC
*	PAYLOAD_OWNER
	PAYLOAD_ORIGIN
*	LAUNCH_DATE
*	LAUNCH_SITE
	MISSION_TYPE
	MISSION_FUNCTION
	MISSION_STATUS

Figure 5: MUC-7 template coverage: asterisks marking all the slots that could be automatically generated.

defined scenario templates quite well. The two MUC7 templates are shown as a list of slots in Figure 5, where horizontal lines delimit slots about different semantic roles, and asterisks mark all the slots that could be automatically generated by our system once it has an improved generalizer. We can see substantial amount of overlap, indicating that a STC system powered by CSC is able to capture scenarios' important facts.

5 Conclusion

We have introduced a new context-sensitive approach to the scenario template creation (STC) problem. Our method leverages deep NL processing, using semantic role labeler's structured semantic tuples as input. Despite the use of deeper semantics, we believe that intrinsic semantic similarity by itself is not sufficient for clustering. We have shown this through examples and argue that an approach that considers contextual similarity is necessary. A key aspect of our work is the incorporation of such contextual information. Our approach uses a notion of context that combines two aspects: positional similarity (when two tuples are adjacent in the text), and argument similarity (when they have similar arguments). The set of relevant articles are represented as graphs where contextual evidence is encoded.

By mapping our problem into a graphical formalism, we cast the STC clustering problem as one of multiple graph alignment. Such a graph alignment is solved by an adaptation of EM, which handles contexts and real-valued similarity by treating both as noisy and potentially unreliable observations.

While scenario template creation (STC) is a difficult problem, its evaluation is arguably more difficult due to the dearth of suitable resources. We have compiled and released a corpus of over 700 newswire articles that describe different instances of 15 scenarios, as a suitable input dataset for further STC research. Using this dataset, we have evaluated and analyzed our context-sensitive approach. While our results are indicative, they show that considering contextual evidence improves performance.

Acknowledgments

The authors are grateful to Kathleen R. McKeown and Elena Filatova at Columbia University for their stimulating discussions and comments over different stages of the preparation of this paper.

References

Robin Collier. 1998. *Automatic Template Creation for Information Extraction*. Ph.D. thesis, University of Sheffield, UK.

A.P. Dempster, N.M. Laird, and D.B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *JRSSB*, 39:1–38.

Elena Filatova, Vasileios Hatzivassiloglou, and Kathleen McKeown. 2006. Automatic creation of domain templates. In *Proceedings of the COLING/ACL '06*.

Sanda M. Harabagiu and V. Finley Lacatusu. 2005. Topic themes for multi-document summarization. In *Proceedings of SIGIR '05*.

Sanda M. Harabagiu and S. J. Maiorano. 2002. Multi-document summarization with GISTEXTER. In *Proceedings of LREC '02*.

Graeme Hirst and Alexander Budanitsky. 2005. Correcting real-word spelling errors by restoring lexical cohesion. *Natural Language Engineering*, 11(1).

Andreas Hotho, Steffen Staab, and Gerd Stumme. 2003. WordNet improves text document clustering. In *Proceedings of the SIGIR 2003 Semantic Web Workshop*.

LDC. 2002. The aquaint corpus of english news text, catalog no. LDC2002t31.

Yoong Keok Lee and Hwee Tou Ng. 2002. An empirical evaluation of knowledge sources and learning algorithms for word sense disambiguation. In *Proceedings of EMNLP '02*.

Dekang Lin. 1998. Automatic retrieval and clustering of similar words. In *Proceedings of COLING/ACL '98*.

Sameer Pradhan, Wayne Ward, Kadri Hacioglu, James Martin, and Dan Jurafsky. 2004. Shallow semantic parsing using support vector machines. In *Proceedings of HLT/NAACL '04*.

Judita Preiss. 2001. Local versus global context for wsd of nouns. In *Proceedings of CLUK4*.

Long Qiu, Min-Yen Kan, and Tat-Seng Chua. 2006. Paraphrase recognition via dissimilarity significance classification. In *Proceedings of EMNLP '06*.

Ellen Riloff and M. Schmelzenbach. 1998. An empirical approach to conceptual case frame acquisition. In *Proceedings of WVLC '98*.

Kiyoshi Sudo, Satoshi Sekine, and Ralph Grishman. 2003. An improved extraction pattern representation model for automatic IE pattern acquisition. In *Proceedings of ACL '03*.

Yuen-Hsien Tseng, Chi-Jen Lin, Hsiu-Han Chen, and Yu-I Lin. 2006. Toward generic title generation for clustered documents. In *Proceedings of AIRS '06*.

Roman Yangarber, Ralph Grishman, Pasi Tapanainen, and Silja Huttunen. 2000. Unsupervised discovery of scenario-level patterns for information extraction. In *Proceedings of ANLP '00*.