Rel-grams: A Probabilistic Model of Relations in Text

Niranjan Balasubramanian, Stephen Soderland, Mausam, and Oren Etzioni
Turing Center, Department of Computer Science and Engineering
Box 352350
University of Washington
Seattle, WA 98195, USA
{niranjan,soderlan,mausam,etzioni}@cs.washington.edu

Abstract

We introduce the Rel-grams language model, which is analogous to an n-grams model, but is computed over relations rather than over words. The model encodes the conditional probability of observing a relational tuple R, given that R' was observed in a window of prior relational tuples. We build a database of Rel-grams co-occurence statistics from ReVerb extractions over 1.8M news wire documents and show that a graphical model based on these statistics is useful for automatically discovering event templates. We make this database freely available and hope it will prove a useful resource for a wide variety of NLP tasks.

1 Introduction

The Google N-grams corpus (Brants and Franz, 2006) has enjoyed immense popularity in NLP and has proven effective for a wide range of applications (Koehn et al., 2007; Bergsma et al., 2009; Lin et al., 2010). However, it is a lexical resource and provides only local, sentence-level information. It does not capture the flow of *semantic* content within a larger document or even in neighboring sentences.

We introduce the novel Rel-grams database[1] containing corpus statistics on frequently occurring sequences of open-domain relational tuples. Rel-grams is analogous to n-grams except that instead of word sequences within a sentence, it tabulates relation sequences within a document. Thus, we expect Rel-grams to model semantic and discourse-level regularities in the English language.

[1] available at *relgrams.cs.washington.edu*.

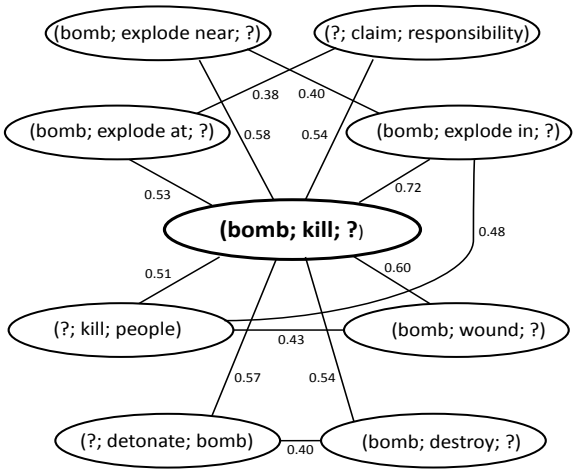

Figure 1: Part of a sub-graph that Rel-grams discovers showing relational tuples strongly associated with *(bomb; kill; ?)*

We have compiled a Rel-grams database from 1.8 million New York Times articles from the Gigaword corpus (Gigaword, 2011). The implementation is linear in the size of the corpus and easily scaled to far larger corpora. Rel-grams database facilitates several tasks including:

Relational Language Models: We define a relational language model, which encodes the probability of relational tuple R, having observed R' in the k previous tuples. This can be used for discourse coherence, sentence order in summarization, *etc.*

Event Template Construction: We cluster commonly co-occuring relational tuples as in Figure 1 and use them as the basis for open event templates (see Table 2). Our work builds on and generalizes earlier efforts by Chambers and Jurafsky (2011).

Expectation-driven Extraction: The probabilities output by the relational language model may be

Proc. of the Joint Workshop on Automatic Knowledge Base Construction & Web-scale Knowledge Extraction (AKBC-WEKEX), pages 101–105,
NAACL-HLT, Montréal, Canada, June 7-8, 2012.

used to inform an information extractor.

As has been the case with n-gram models and resources such as DIRT (Lin and Pantel, 2001), we expect the community to suggest additional applications leveraging this large scale, public resource. An intriguing possibility is to use Rel-grams in document-level extraction or summarization to assess the discourse coherence of alternate hypotheses in a decoding step in much the same way that n-grams have been used in speech or statistical MT.

2 Rel-grams & Relational Language Model

We build a relational language model that specifies the probability of observing the next relational tuple in a sequence of tuples. As an approximation, we estimate bi-gram probabilities. Formally, we use $P_k(R|R')$ as the probability that R follows R' within a distance of k tuples, as a delta-smoothed estimate:

$$P_k(R|R') = \frac{\#(R, R', k) + \delta}{\#R' + \delta \cdot |V|} \quad (1)$$

where, $\#(R, R', k)$ is the number of times R follows R' in a document within a distance of k tuples. $k = 1$ indicates consecutive tuples in the document. $\#R'$ is the number of times R' was observed in the corpus. $|V|$ is the number of unique tuples in the corpus. Notice that, similar to typical language models, this is a sequential model, though we can define undirected models too.

We can use inference over the relational language model to answer a variety of questions. As an example, we can compute the semantic coherence of a document D by converting it into a sequence of relational tuples $< R_1, ..., R_n >$ and computing the joint probability of observing the document D:

$$P(D) = P_{seq}(< R_1, ..., R_n >) \quad (2)$$

$$= P(R_1) \prod_{i=2}^{n} P_1(R_i|R_{i-1}) \quad (3)$$

This can be used to score alternate sequences of tuples in a decoding step, ranking the coherence of alternate versions of a generated document or summary.

Another use of the relational language model is to assess the likelihood of a tuple R given a set of tuples within a window of k tuples. This can serve as a verification during NLP tasks such as extraction.

Table 1: Generalized tuples with the highest conditional probability given *(treasury bond; fall; ?)*. Our model matches well with human intuition about semantic relatedness.

Predicted tuples	Argument values
1.(bond; fall; ?)	point, percent
2.(yield; rise; ?)	point, percent
3.(report; show; ?)	economy, growth
4.(bond; yield; ?)	percent, point
5.(index; rise; ?)	percent, point
6.(federal reserve; raise; ?)	rate, interest rate

2.1 The Rel-grams Database

As a first step towards building the relational language model, we extract relational tuples from each sentence in our corpus using ReVerb, a state-of-the-art Open IE system (Fader et al., 2011). This extracts relational tuples in the format (arg1; rel; arg2) where each tuple element is a phrase from the sentence. We construct a relational database to hold co-occurrence statistics for pairs of tuples found in each document as shown in Figure 2. The database consists of three tables: a *Tuples* table maps each tuple to a unique identifier; *BigramCounts* stores the co-occurrence frequency, a count for a pair of tuples and a window k; *UniCounts* counts the number of times each tuple was observed in the corpus. We refer to this resource as the Rel-grams database.

Query Language: The relational nature of the Rel-grams database allows for powerful querying using SQL. For example, the database can be used to find the most frequent Rel-grams, whose first tuple has *bomb* as the head of its arg1 and has *explode near* as its predicate (with no constraints on arg2). We find that the views in which one or more of the elements in a tuple is a wild-card, are often useful (as in this example). We call these generalized tuples. We materialize tables for generalized tuples in advance to improve query performance.

The Rel-grams database aggregates important information regarding the occurence of semantic relations in documents and allows general querying capability. We envision the database to be also useful for several other tasks such as building event templates (Section 3).

We populated the Rel-grams database using ReVerb extractions from a subset of 1.8 million New York Times articles from the Gigaword corpus. To reduce sparsity, we represented the arguments and

Tuples

Id	Arg1Head	RelNorm	Arg2Head
...	...	...	...
...	...	...	...
13	bomb	explode near	market
...	...	...	...
20	bomb	kill	people
...	...	...	...
...	...	...	...

UniCounts

Id	Count
...	..
...	..
13	45
...	..
20	37
...	..
...	..

BigramCounts

T1	T2	Window	Count
...	...	...	...
13	20	3	12
13	20	9	16
...	...	...	...
20	13	5	2
20	13	7	5
...	...	...	...

Figure 2: Rel-grams Database

relation phrases by their stemmed head words, including prepositions for the relation phrases. The database consisted of over 97M entries.

2.2 Evaluation

First, we evaluated the relational language model to test if the conditional probabilities $P_k(R'|R)$ (Equation 1) reflect semantic relatedness between tuples. We took 20 arbitrary generalized tuples R' and obtained the top 50 tuples $\{R\}$ with highest conditional probability $P_5(R|R')$. Annotators considered each R to be correct if there was a close semantic relation with R', not necessarily synonymy. We found high precision, over 0.96, for the top 50 predicted tuples. Table 1 shows the top few tuples associated with the generalized tuple *(treasury bond; fall; ?)*. Notice that including an argument adds essential information to otherwise vague relation phrase such as "fall", "rise", "show", or "yield".

Next, we evaluated the relational language model on a pseudo-disambiguation task: distinguishing between randomly generated documents and original news articles. We created a test set as follows. From a random sample of the AFP portion of the Gigaword corpus, we selected the first ten documents that covered a non-redundant set of topics such as sports, politics, etc. Then, we pooled the sentences from these ten documents and built pseudo-documents by randomly sampling sentences from this pool.

Given a document, D, we extract the sequence of relational tuples $< R_1, ..., R_n >$. Then, we compute the likelihood of observing this sequence as shown in Equation 2 with window size k set to 5. To account for sparsity, we employ back-off models that switch to conditional probabilities of generalized tuples. We normalize the likelihoods to account for different length sequences.

The average likelihood of original news articles was 2.5 times higher than that of pseudo-documents created from a randomly sampled sentences. This suggests that relational language model captures important semantic information.

3 Event Template Discovery

The Rel-grams database can also be used to identify groups of tuples that frequently co-occur with each other in a specific event or scenario. In particular, we are motivated by the task of automatically building event templates. We cast this a clustering problem on a graph of relation tuples (Rel-graphs).

3.1 Rel-graphs

We define Rel-graphs as an undirected weighted graph $G = (V, E)$, whose vertices (V) are generalized relation tuples, and whose weighted edges (E) represent the strength of co-occurrences between each pair of tuples. We generalize each relation tuple by replacing either argument with a wild-card obtaining (arg1; rel; ?) and (?; rel; arg2). We then create edges for all pairs of these generalized tuples that co-occurred at least five times in the Rel-gram database. To assign edge weights, we choose normalized point-wise mutual information (PMI), which is computed as follows:

$$PMI(R, R') = \log \left\{ \frac{\#(R, R', k) + \#(R', R, k)}{\#R \, \#R'} \right\}$$

where, $\#(R, R', k) + \#(R', R, k)$ is the number of times R and R' co-occur within a window of k, which we set to 10 for this experiment. We normalize the PMI score by the maximum marginal probability of the tuples. The resulting graph consisted of more than 320K vertices and more than 2M edges.

3.2 Event Templates from Clustering

Tightly connected clusters on the Rel-graphs represent frequently co-occurring tuples. These clusters may be viewed as representing event templates,

Table 2: A sample of the 50 highest connectivity Rel-clusters. We show only the first few nodes of each cluster and the highest frequency argument values for the open slot. About 89% of nodes in these 50 clusters are relevant.

Top Nodes	Arguments	Top Nodes	Arguments	Top Nodes	Arguments
(suit; seek; ?)	damages, status	(disease; cause ; ?)	death, virus	(sale; increase; ?)	percent, year
(lawsuit; file in; ?)	court, state	(study; show ; ?)	drug, people	(sale; account; ?)	revenue,sale
(case; involve; ?)	woman, company	(people; die; ?)	year, disease	(profit; rise; ?)	percent, pound
(suit; accuse; ?)	company, Microsoft	(disease; kill; ?)	people, woman	(share; fall; ?)	percent, cent
(?; file; suit)	group, lawyer	(?; treat; disease)	drug, cell	(share; rise; ?)	percent, penny
(suit; allege; ?)	company, fraud	(?; kill; people)	bomb, attack	(analyst; survey by; ?)	bloomberg,zacks
Top Nodes	**Argument Values**	**Top Nodes**	**Argument Values**	**Top Nodes**	**Argument Values**
(film; win; ?)	award, prize	(state; allow ; ?)	doctor, company	(patriot; play; ?)	game, sunday
(film; receive; ?)	review, rating	(law; require ; ?)	company, state	(patriot; in; ?)	game,league
(film; earn; ?)	year, million	(state; require; ?)	company, student	(patriot; win; ?)	game, super bowl
(?; make; film)	director, studio	(state; pass; ?)	law, legislation	(patriot; get; ?)	break, player
(film; get; ?)	nomination, review	(state; use; ?)	money, fund	(patriot; lose; ?)	game, sunday
(?; see; film)	people, anyone	(?; require; state)	bill, Congress	(?; rush; yard)	smith, williams

where the arguments are the entities (slots) in the event and the relation phrase represents their roles.

We employed Markov clustering (Van Dongen, 2008) to find tightly connected clusters on the Rel-graphs. Markov clustering finds clusters efficiently by simulating random walks on the graph.[2] The clustering produced 37,210 clusters for our Rel-graph, which we refer to as Rel-clusters.

We observed that our clusters often included tuples that were only weakly connected to other tuples in the cluster. To prune unrelated tuples, we devise a two step process. First, we select the top three most connected vertices within the cluster. Starting with these three vertices, we compute a subgraph including the direct neighbors and all their pairwise edges. We then sort the vertices in this sub-graph based on their total edge weights and select the top 50 vertices. Figure 1 shows a portion of such a cluster, with vertices strongly connected to (bomb; kill; ?) and all edges between those vertices.

Table 2 shows the top nodes for a sample of high connectivity Rel-clusterswith the two most frequent argument values for their wildcard slot.

3.3 Evaluation

First we evaluated the semantic cohesiveness of a random sample of the 50 clusters with highest connectivity. We found that about 89% of the nodes in each cluster were semantically related to the implicit topic of the cluster.

Next, we evaluate Rel-clusters with an independent gold standard. We compare against MUC-4 templates for terrorist events: bombing, attack, kidnapping, and arson. MUC-4 templates have six primary extraction slots – perpetrator, victim, physical target (omitted for kidnapping), instrument (omitted for kidnapping and arson), date, and location.

To obtain Rel-clusters for these four terrorist event types, we look for clusters that include the seed extractions: *(bomb; explode; ?), (attack; kill; ?), (?; kidnap; ?), (?; set fire; ?)*. We examine the argument values for these nodes to see whether the argument type corresponds to a slot in the MUC-4 event template and use it to compute recall.

Table 3 shows the performance our Rel-clusters and compares it with the MUC-4 template slots discovered by an unsupervised template extraction approach (Chambers and Jurafsky, 2011). We find that Rel-clusters were able to find a node with arguments for all six slots for bombing and attack event types. It had more difficulty with kidnapping and arson, missing the date and location for kidnapping and missing the victim and location for arson. Chambers missed one victim and did not include date or location for any template.

We view these as promising preliminary results but do not draw any strong conclusions on the comparison with Chambers and Jurafsky, as unlike our system, theirs was designed to produce not only templates, but also extractors for the slots.

In the future, we will automatically determine semantic types for the slots. We will also split slots that have a mixture of semantic types, as in the example of the arguments {percent, year} for the extraction *(sale; increase; ?)* in Table 2.

[2]We use an efficient sparse matrix implementation from `http://micans.org/mcl/` that scales linearly in the number of graph vertices.

Table 3: Both Rel-clusters and Chambers system discovered clusters that covered most of the extraction slots for MUC-4 terrorism events.

	Fraction of slots	
	Chambers	Rel-clusters
Bombing	0.50	1.00
Attack	0.67	1.00
Kidnapping	0.50	0.50
Arson	0.60	0.60
Average	0.57	0.77

4 Related Work

There has been extensive use of n-grams to model language at the word level (Brown et al., 1992; Bergsma et al., 2009; Momtazi and Klakow, 2009; Yu et al., 2007; Lin et al., 2010). Rel-grams model language at the level of relations. Unlike DIRT (Lin and Pantel, 2001), Rel-grams counts relation co-occurrence rather than argument co-occurence. And unlike VerbOcean (Chklovski and Pantel, 2004), Rel-grams handles arbitrary relations rather than a small set of pre-determined relations between verbs.

We build on prior work that learns narrative chains and narrative schema that link actions by the same protagonists (Chambers and Jurafsky, 2008; Chambers and Jurafsky, 2009), and work that extracts event templates from a narrowly focused corpus (Chambers and Jurafsky, 2011). Rel-grams finds more general associations between relations, and has made a first step towards learning event templates at scale.

5 Conclusions

This paper introduces the Rel-grams model, which is analogous to n-gram language models, but is computed over relations rather than over words. We construct the Rel-grams probabilistic graphical model based on statistics stored in the Rel-grams database and demonstrate the model's use in identifying event templates from clusters of co-occurring relational tuples. The Rel-grams database is available to the research community and may prove useful for a wide range of NLP applications.

6 Acknowledgements

This research was supported in part by NSF grant IIS-0803481, ONR grant N00014-08-1-0431, and DARPA contract FA8750-09-C-0179, and carried out at the University of Washington's Turing Center.

References

S. Bergsma, D. Lin, and R. Goebel. 2009. Web-scale N-gram models for lexical disambiguation. In *Proceedings of IJCAI*.

Thorsten Brants and Alex Franz. 2006. The Google Web1T 5-gram Corpus Version 1.1. LDC2006T13.

P. Brown, P. deSouza, R. Mercer, V. Della Pietra, and J. Lai. 1992. Class-based n-gram models of natural language. *Computational Linguistics*.

N. Chambers and D. Jurafsky. 2008. Unsupervised learning of narrative event chains. In *Proceedings of ACL-08: HLT*.

N. Chambers and D. Jurafsky. 2009. Unsupervised learning of narrative schemas and their participants. In *Proceedings of ACL*.

N. Chambers and D. Jurafsky. 2011. Template-based information extraction without the templates. In *Proceedings of ACL*.

T. Chklovski and P. Pantel. 2004. VerbOcean: Mining the web for fine-grained semantic verb relations. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-04)*, pages 33–40.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of EMNLP*.

English Gigaword. 2011. http://www.ldc.upenn.edu/Catalog/catalogEntry.jsp?catalogId=LDC2011T07.

P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst. 2007. Moses: open source toolkit for statistical machine translation. In *Proceedings of ACL*.

D. Lin and P. Pantel. 2001. DIRT – Discovery of Inference Rules from Text. In *Proceedings of KDD*.

D. Lin, K. Church, H. Ji, S. Sekine, D. Yarowsky, S. Bergsma, K. Patil, E. Pitler, R. Lathbury, V. Rao, K. Dalwani, and S. Narsale. 2010. New tools for Web-scale N-grams. In *Proceedings of LREC*.

S. Momtazi and D. Klakow. 2009. A word clustering approach for language model-based sentence retrieval in question answering systems. In *Proceedings of CIKM*.

S. Van Dongen. 2008. Graph clustering via a discrete uncoupling process. *SIAM Journal on Matrix Analysis and Applications*, 30(1):121–141.

L-C. Yu, C-H. Wu, A. Philpot, and E. Hovy. 2007. OntoNotes: sense pool verification using Google N-gram and statistical tests. In *Proceedings of OntoLex Workshop*.