The UTexas system for TAC SM-KBP task 3: Probabilistic generation of coherent hypotheses

Pengxiang Cheng **Eric Holgate** **Katrin Erk**
University of Texas at Austin
{pxcheng, holgate}@utexas.edu katrin.erk@mail.utexas.edu

1 Introduction

As an event unfolds, such as a crime, natural disaster, or conflict, it is often initially unclear what happened. Information is spread over many sources in different modalities and languages, and is often conflicting. For example, when flight MH-17 crashed in Ukraine in 2014, there were initially many theories of what happened, including a missile strike initiated by Russia-affiliated militants, a missile strike by the Ukrainian military, and a terrorist attack.

Streaming Multimedia Knowledge Base Population (SM-KBP) is a new track at TAC 2018 that addresses this scenario through multiple tasks. Task 1 is about the extraction of information from multimodal and multi-lingual sources. Task 2 combines document-level knowledge graphs into a single multi-document knowledge graph and performs cross-document coreference. The resulting knowledge graph will contain incompatible narratives on what happened. Task 3 constructs "hypotheses", internally consistent narratives, out of the overall incompatible knowledge base. Task 3 can thus be viewed as a "worst case" of multi-document summarization (Christensen et al., 2013; Wities et al., 2017) in which the documents tell incompatible stories.

The UTexas team participated in Task 3, hypothesis generation. We view hypothesis generation as a clustering task based on narrative coherence (Wang et al., 2018). However, it is not a standard clustering task, but clustering combined with inference, where the inference incorporates background knowledge on the compatibility or incompatibility of knowledge subgraphs. The UTexas system for TAC 2018 uses probabilistic programming (Goodman and Stuhlmüller, 2014) to implement a probabilistic generative process that selects connected and compatible subgraphs.

2 The Hypothesis Generation Task

The input to SM-KBP task 3, hypothesis generation, was a knowledge graph generated by task 2. This graph was given in AIDA Interchange Format (AIF), an RDF representation of the knowledge graph in turtle format.

AIF nodes characterize entities, events, and relations – that is, events as well as relations are reified. AIF statements describe the types of entities, events and relations, as well as links between them. The labels used in these statements are specified in the AIDA Seedling Ontology. Coreference between entities, between events, or between relations is also described in the knowledge graph. SameAsCluster nodes introduce coreference groups, and membership in a coreference group is stated by a ClusterMembership statement. All statements, including coreference statements come with confidence weights.

The hypotheses to be generated are characterized through *Statements of Information Need*. Such a Statement includes entities, events, or relations to be included in the hypothesis, called *entry points*. As the underlying data is multimodal, an entry point may be characterized through text (for example, an entity's name), an image, or a piece of a video. Additionally, a Statement of Information Need specifies frames: different takes on a situation. Returning again to the MH-17 example, a single entry point describing flight MH-17 may be mentioned in two frames, one that states that the event was an accident, the other stating that it was an attack. Each frame is given in the form of labeled edges. A third important piece of information in the Statement of Information Need is a number of *hops*, which specifies how far

in the graph to go from the entry points to find possible members of a hypothesis. By NIST's definition of hops, two coreferent nodes are zero-hop neighbors; two nodes connected by a role edge are a half-hop apart, and all other statements (such as type statements) count for one hop.

Task 3 of SM-KBP had two parts. The first subtask was to generate subgraphs that meet a given Statement of Information Need, selecting only from the nodes of the knowledge graph that are at most the given number of hops away from an entry point. The second subtask was to execute queries over the constructed hypothesis subgraphs. Queries ask about particular knowledge elements in the given subgraph.

Incompatibility between hypotheses is introduced through incompatible statements – for example, both Russia-affiliated militants and the Ukrainian military may be stated as `Agents` of an event of type `Conflict.Attack`. As mentioned above, we view hypothesis generation as a clustering task. As hypotheses need to differ in the statements they include, the elements that we cluster are AIF statements.

3 The UTexas system

3.1 Probabilistic hypothesis generation using probabilistic programming

Outline. We view hypothesis generation as a clustering task (Wang et al., 2018), [1] or more specifically, an instance of one-class clustering (Crammer and Chechik, 2004; Gupta and Ghosh, 2005; Bekkerman and Crammer, 2008). In one-class clustering, as in one-class classification, we have only *one* class of interest; all other datapoints are considered outliers. The difference between one-class classification and one-class clustering is that while one-class classification assumes that outliers are rare, one-class clustering aims to extract a subset of closely linked relevant datapoints in a space with many outliers. In our case, the cluster we aim to extract is a hypothesis.

We generate a hypothesis through a probabilistic generative process. The process starts from an initial cluster from entry points as identified in a Statement of Information Need. We then use a particle filter or sequential Monte Carlo method to repeatedly sample statements to add to the cluster, based on their degree of coherence with the cluster generated so far. After each extension of the cluster, we check for coherence through hard and soft logical rules. We implement this probabilistic generative process in the probabilistic programming language WebPPL (Goodman and Stuhlmüller, 2014). Each sample generated in this way constitutes a hypothesis. We keep the n samples with the highest probability and return them as the system-generated hypotheses.

After this outline of the process, we now describe more details.

Statement sampling. The one-class cluster that is built by the generative process consists of statements. A statement is considered as a candidate for extending the cluster only if it mentions an entity, event, or relation that is already mentioned in the cluster. This ensures that the generated hypothesis is a connected subgraph. At the beginning of cluster generation, the process samples a random frame from the Statement of Information Need. Then cluster generation proceeds in two stages. The first stage seeks to satisfy the edges specified in the frame: For each frame edge, a random candidate statement is sampled from all the statements that would satisfy the edge, and consistency is checked using logical rules.

The second stage adds statements not specified in the frame. In this stage, cluster membership of a sampled statement is determined probabilistically based on a fuzzy boundary on coherence with the cluster. For each candidate statement, the fuzzy boundary is sampled from a gamma distribution (whose shape and scale are hyperparameters of the model). If the coherence of the candidate statement to the current cluster exceeds the boundary, it is added to the cluster, and again consistency is checked using logical rules. In the current model, we use random walk proximity within the knowledge graph as the measure of coherence.

Coreference. We model coreference grouping as probabilistic unification, where an entity, event, or relation e is unified with the prototype member of a coreference group g with a probability that depends

[1] In the AIDA Interchange Format, the term "cluster" is used to refer to groups of coreferent entities. Here and in the following, we use the term "clustering" to refer to a method for hypothesis generation, and refer to coreferent entities as a "coreference group".

on the coreference confidence of e and g as stated in the knowledge graph.

Logical rules. Hard and soft logical rules can be specified as hard and soft sampling constraints in WebPPL. A hard constraint, if violated, causes a sample to be rejected. The weight of a soft constraint is added to the log probability of the sample. (1) shows an example of a hard constraint, specified as a `condition`. This constraint states that a node cannot have two conflicting types: If the subjects of two type statements are the same modulo coreference, then the objects have to be coreferent too.

$$\begin{array}{ll} condition(& !unifEqual(typestmt1.subject, typestmt2.subject) \ || \\ & unifEqual(typestmt1.object, typestmt2.object)) \end{array} \tag{1}$$

(2) is an example of a soft constraint, specified as a `factor`. For predicates that should be unique in their objects, such as `Conflict.Attack_Agent`, a sample takes a penalty of $-weight$ if it has two such statements with the same subject and different objects.

$$\begin{array}{lll} factor(& !uniquePred(stmt1.predicate) & || \\ & stmt1.predicate != stmt2.predicate & || \\ & !unifEqual(stmt1.subject, stmt2.subject) & || \\ & unifEqual(stmt1.object, stmt2.object) & \\ & ? \ 0 \ : \ -weight) & \end{array} \tag{2}$$

This mechanism is closely related to Markov Logic Networks (Richardson and Domingos, 2006). When we view a sample as a truth assignment or possible world, a soft constraint specifies a bonus given to that assignment when the constraint is satisfied.

3.2 The UTexas pipeline

The input to our system is provided by a task 2 output. We used the output from the Colorado Ramfis team, which in turn was constructed from the task 1 output of the GAIA team.

Our system consists of the following components. Components 1-3 are preprocessing steps, component 4 does the actual hypothesis generation, and components 5-7 perform postprocessing.

1. Statement of Information Need processing. We use Apache Jena to query and manipulate the knowledge graph. In this first preprocessing step, the input knowledge graph is loaded into a TDB database, the *whole-graph TDB*. Given a Statement of Information Need, we use SPARQL queries to extract the best match for each entry point. This step may induce additional coreference statements: If a frame states that an entry point should participate in an edge labeled η but the best match e for that entry point does not participate in any edge labeled η, we unify e with the best matching node with an η edge.

2. k-hop subgraph extraction. We use a Python script to identify zero-hop, half-hop, and one-hop neighbors of each node. This is feasible because we first break up the very large input .ttl file into smaller files of manageable size. Also, the neighbor identification process is linear with respect to the number of nodes in the graph.

Using the best entry point matches determined during Statement of Information Need processing, we generate SPARQL query rules that we use on the whole-graph TDB to extract all nodes that are up to k hops away form an entry point. The result is a second TDB, the *subgraph TDB*. We also transform the subgraph into a json format for input to WebPPL.

3. Coreference reduction. We use a Python script to perform probabilistic unification for coreference. This step operates on the json version of the subgraph. This step can be performed multiple times to obtain multiple coreference samples; in the current evaluation, we only performed this step once for each subgraph. The probabilistic unification yields a partition of all entities, events, and relations into coreference equivalence classes. We rewrite the subgraph modulo coreference equivalence, which drastically reduces the size of the subgraph.

4. Hypothesis generation. To estimate the shape and scale of the gamma distribution for the fuzzy threshold, we observe neighbor proximity in the subgraph for some sampled connections, and set shape and scale such that the mean of the gamma distribution isc lose to the mean observed neighbor proximity.

Then we run the WebPPL module for probabilistic cluster generation, and retain the n samples with the highest probability as the generated clusters.

5. Coreference expansion. We re-write the output of the WebPPL module to transform nodes that are coreference equivalence classes back to the original entities, events, and relations, with the appropriate coreference statements to link them. The output of this step is a re-expanded hypothesis graph in json format.

6. Output format generation. For each hypothesis, we formulate a set of SPARQL update rules to create a modified version of the subgraph TDB that omits all triples that are not in the hypothesis, and that includes an aida:Hypothesis wrapper. The resulting subgraph is written to a ttl file.

7. Query execution. We use a tool from the Colorado *Ramfis* team to apply the NIST-provided queries to the hypothesis subgraphs that we construct. This tool applies SPARQL queries to TDB objects.

3.3 System retooling during the evaluation period

At roughly 34G, the input file that we obtained was substantially larger than the data that we had previously worked with. The Statements of Information Need were also substantially larger than examples seen before. As a result, we needed to modify all parts of our system.

Before the evaluation period, our pipeline used Python for all steps of the pre- and postprocessing. This turned out not to be feasible with the evaluation data, so we switched several components to using Apache Jena. We also added the k-hop subgraph extraction step to reduce the amount of data handed to downstream components.

The probabilistic unification for coreference handling was originally included in the core WebPPL module. However, during the evaluation we found that WebPPL was too slow when working on subgraphs that were not coreference-reduced. So we re-implemented coreference handling in a separate Python module, which again considerably reduced the size of the subgraphs handed to downstream components. The disadvantage of this setup is that the evaluation system currently does not support joint inference over coreference and narrative coherence.

For query execution, we initially used the SM-KBP 2018 Evaluation Queries Docker tool[2] provided by NIST. However, this tool did not scale up well to the large size of the hypothesis subgraphs generated by our system. So we switched to a tool provided by the Colorado Ramfis team, as mentioned above.

4 Discussion

Our pipeline, with its mixture of Jena queries, Python scripts, and probabilistic programs, is currently too complex and needs too much oversight. It will need to be simplified and scaled up for future evaluations. On the positive side, we found that Apache Jena enabled us to deal even with the large amount of data given in this evaluation. Also the data reduction methods we implemented, k-hop subgraph extraction and coreference reduction, made our system much more scalable.

The system submitted by the UTexas team is a baseline system that is completely symbolic. The eventual aim of the project is to use machine learning to learn narrative coherence from large amounts of text. This notion of coherence would then replace the baseline random-walk proximity that we have been using in this evaluation. However there was no training data in the form of knowledge graphs available prior to the evaluation. At this point, after the evaluation, large-scale training data analyzed by task-1 and task-2 teams is becoming available, such that in the future we will be able to explore machine learning models as planned.

Which inference should be done within which task? This is a fundamental question that needs to be discussed in the future. For example, is coreference something that should be inferred along with

[2] `https://tac.nist.gov/protected/2018-kbp/data/SM-KBP_2018_Evaluation_Queries_Docker.tgz`

logical consistency in task 3, or is this better handled within task 2? The latter approach would follow the AIDA principle of allowing for inference in every task, with information flowing between tasks in both directions. Also, this would allow task 3 to operate on coreference-reduced data for more scalable inference.

Acknowledgements

This research was supported by the DARPA AIDA program under AFRL grant FA8750-18-2-0017. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of DARPA, DoD or the US government. We acknowledge the Texas Advanced Computing Center for providing grid resources that contributed to these results. Results presented in this paper were obtained using the Chameleon testbed supported by the National Science Foundation.

References

Ron Bekkerman and Koby Crammer. 2008. One-class clustering in the text domain. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Janara Christensen, Mausam, Stephen Soderland, and Oren Etzioni. 2013. Towards coherent multi-document summarization. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Koby Crammer and Gal Chechik. 2004. A needle in a haystack: local one-class optimization. In *Proceedings of ICML.*

Noah D Goodman and Andreas Stuhlmüller. 2014. The design and implementation of probabilistic programming languages. `http://dippl.org`.

Gunjan Gupta and Joydeep Ghosh. 2005. Robust one-class clustering using hybrid global and local search. In *Proceedings of ICML.*

Matthew Richardson and Pedro Domingos. 2006. Markov logic networks. *Machine Learning*, 62(1):107–136.

Su Wang, Eric Holgate, Greg Durrett, and Katrin Erk. 2018. Picking apart story salads. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium.

Rachel Wities, Vered Shwartz, Gabriel Stanovsky, Meni Adler, Ori Shapira, Shyam Upadhyay, Dan Roth, Eugenio Martinez Camara, Iryna Gurevych, and Ido Dagan. 2017. A consolidated open knowledge representation for multiple texts. In *Proceedings of the 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics*, Valencia, Spain.