NIST TAC SM-KBP 2018
System Description: JHU/UR Pipeline

Patrick Xia and **Elias Stengel-Eskin** and **Tongfei Chen**
Seth Ebner and **Nils Holzenberger** and **Ryan Culkin**
Pushpendre Rastogi and **Xutai Ma** and **Benjamin Van Durme**

Johns Hopkins University

Abstract

We constructed a pipeline-based system for participation in the 2018 pilot NIST SM-KBP evaluation. Our pipeline was assembled from a series of components representing contemporary approaches to each of the distinct Information Extraction and Machine Translation tasks required. Our goals for this system were: (1) to demonstrate an extant SM-KBP capability (can we run on the data and produce a valid output); and (2) to stand as a non-trivial baseline for contrasting against our future efforts for this task.

1 Introduction

The SM-KBP task involved processing a set of independent documents in English, Russian, and Ukrainian. For each document we were required to identify named entities, coreferring expressions, and events and relations the entities appeared in. Under the guidelines for the task, evidence for events and relations may occur in different sentences of a document than the supporting mention(s) for an entity that is deemed to be a relational argument or event participant. The task was evaluated via queries consisting of offsets into a given document, representing oracle span annotations for some initial element salient to the query. Based on a query, various knowledge structures were to be returned, represented in the AIDA Interchange Format (AIF). Task 1a consisted of a corpus and queries; Task 1b consisted of a corpus, queries, and associated hypothesis objects that could viewed as partial knowledge base fragments, upon which the extraction system should condition during document processing.

The 2018 instance of SM-KBP was considered a pilot, with participation by TA1 performers of DARPA's AIDA program. Our team constructed a system meant to establish a baseline performance, with our primary goal being to ensure we gained an understanding of the task, and had a non-trivial baseline in future work.

Our system involved training ERE extraction components on annotated documents provided under the DARPA AIDA and DEFT programs. Coreference was handled by a pre-trained, 3rd party system. No attempt was made to perform extraction directly on non-English documents: these were translated into English, processed with English-based components, then aligned heuristically back to Ukrainian and Russian spans in order to handle source-document queries.

The hypothesis information under Task 1b was handled by extracting named entity strings from the hypothesis, then performing a limited sort of entity linking against those entity strings and the coreference chains extracted under Task 1a. For any potential match, we then aimed to increase the recall of the relations and events we output for those entities. In short: if something downstream had a hypothesis about a particular entity, and we thought a given document could be about that entity, then we increased the amount of information we output, specific to that entity.

The following gives details on the modules involved in this system.

2 Machine Translation

We restricted our extraction models to English, with a reliance on Machine Translation (MT) into English to support non-English documents. We assembled various bitexts for training and interacted with NIST to establish whether each was constrained or unconstrained.

2.1 Model

Translation was performed using the Python implementation of OpenNMT (Klein et al., 2017). A 4-layer bi-directional LSTM encoder-decoder model with attention was employed. The dimensions of the embedding layers and LSTM hidden units were 300 and 1024 respectively.

The following describes the resources used for each language.

2.2 Russian - English

Two corpora were used to build Russian - English MT systems. The first consists of the training and development set from the WMT'17 news translation challenge.[1] It is an aggregate of the Common Crawl corpus, News Commentary v12, Yandex Corpus, Wiki Headlines, and UN Parallel Corpus V1.0, and totals 25M sentences. The UN parallel Corpus mentions Crimea in several places, and so may only be used in the unconstrained setting. The second corpus is the OpenSubtitles 2018 Russian-English corpus [2]. This is parallel text containing 26M sentences extracted from movie subtitles (Lison and Tiedemann, 2016). It does not mention the Crimea conflict and may thus be used in the constrained setting.

2.3 Ukrainian - English

To build Ukrainian - English translation systems, we used the OpenSubtitles 2018 corpus [3], which contains 878k sentences. It was built in the same way as the Russian-English Open Subtitles corpus and can also be used in the constrained setting.

2.4 Language Identification

LDC provided initial, automatic Language Identification (LID) determinations, which on inspection we determined to be error-prone. We therefore performed an internal, sentence by sentence LID analysis via the python port of Google's language-detection library[4]. For each document, the LID system only detected Russian and Ukrainian sentences. A sentence will be categorized as English if it was not classified as Russian or Ukrainian by the LID system. We then translated sentences classified as Russian and Ukrainian with machine translation describe in previous sections.

3 Information Extraction

There are two phases in extracting information from text: (1) independently detect entities, relations, and events; then (2) link detected entities as potential arguments to predicted relations and events.

3.1 Entity Mention and Type Detection

To detect entity mentions, we implemented a neural biLSTM-CRF BIO tagger (Lample et al., 2016), with tokens embedded using 300-dimensional GloVe word embeddings. We trained on ERE data made available under the DEFT program as it contains annotations for not just entities but also entity type, relations, and events. It also is exhaustively annotated (no false negatives), unlike initial data provided under AIDA. In the cases of overlapping spans, we discard all but the longest span. At test time, the tagger is run at the sentence level independently for each sentence, effectively placing each token into at most one span.

Our BIO tags were augmented with entity type labels (according to the DEFT ontology), including filler types. This greatly increased the label space but also allows the model to jointly make decisions regarding entity detection and type prediction.

We experimented with several hyperparameters, such as the number of layers, dropout, and tag types (IO, BIO, BIOE), selecting our final configuration based on a development set from DEFT data. A single-layer BIOE tagger with 0.9 dropout, and l_2 regularization of 0.001 over all the parameters was found to be the most effective, achieving an accuracy of 83.6%, precision of 65.3%, recall of 54.6% and F1 score of 59.5%. While these results are low for state-of-the-art NER, we kept with this solution for the pilot in order to stick with data most similar to the AIDA domains, and with a label set that could be directly mapped (as the filler types are comparable). In addition, manual inspection of the predictions concluded that while the type may be predicted incorrectly, the mention spans were quite accurate. Furthermore, early errors made during entity typing can be corrected later based on document-level information (and coreference resolution). For these reasons, we decided to use this in-domain BIOE tagger for entity mention and type prediction.

3.2 Relation and Event Detection

Our information extraction systems run on English text, some of which may be noisily translated, but graph queries specify spans in source documents. Because we don't have alignments in our training data, we detect the presence of relations and events (REs) at the sentence level instead of at the token level. The span for a found RE is then just the entire sentence. Analysis of the AIDA data showed that most sentences have at most 1 RE mention, so at query time we can find the RE in question simply by finding which sentence a query refers to.

Sentences may contain many REs (rarely of the same type), so we detect each type with its own binary classifier indicating whether an RE of a given type is present in the sentence. Most RE types do not have many labeled examples in the English portion of the AIDA training corpus. We did not incorporate DEFT ERE data as training data for RE detection because the style (forum posts) is too different from that of the AIDA data (profes-

[1] http://data.statmt.org/wmt17/translation-task/preprocessed/ru-en

[2] http://opus.nlpl.eu/download.php?f=OpenSubtitles2018/en-ru.txt.zip

[3] http://opus.nlpl.eu/download.php?f=OpenSubtitles2018/en-uk.txt.zip

[4] https://github.com/Mimino666/langdetect

sional newswire), even though the ontologies are very similar. Our system does not make predictions for types that have no labeled instances.

We address the data sparsity problem in three ways: 1) we use pre-trained GloVe word embeddings (Pennington et al., 2014)[5], giving randomly initialized word embeddings to words appearing in the training data without pre-trained embeddings; 2) we use a simple encoder consisting of mean-pooling concatenated with max-pooling over the sentence's word embeddings, which is then fed through a fully connected layer and a ReLU activation; and 3) we equip each binary RE classifier with a tunable hyperparameter that adjusts the precision and recall of detecting its corresponding RE type. We have found adaptive thresholding to improve performance in related tasks (Zhang et al., 2018), and in practice we adjust the thresholds to avoid the proliferation or lack of predicted types.

These three techniques focus on building tunable models with small amounts of parameters, owing to the lack of data to support larger models. For example, we do not update pre-trained word embeddings during training, which substantially reduces the number of trainable parameters in our models. The encoder we use has been found to be useful in unpublished work in sentence-level topic identification, which is similar to our sentence-level RE detection task. Preliminary experiments with multilabel RE classification resulted in most sentences receiving similar predictions and did not give fine control over model predictions (due to the loss function we used). Our decision to switch to constructing a separate classifier and a tunable precision-recall hyperparameter for each RE type allowed us to qualitatively check that appropriate predictions are made for each type. The hyperparameters can be adjusted at inference time without needing to retrain the model.

3.3 Argument Linking

Once we have predicted an event or relation is triggered in a given sentence, and we have separately detected mentions (with types), then we optionally link each mention to a relation or event as an argument, as a second round of sequence tagging. We use the same framework was used for entity mention and type detection; a biLSTM-CRF tagger. We trained a single model for Events and one for Relations, using an IO tagset (owing to label sparsity in training we forwent a 'B' tag, trusting that very few distinct arguments would be contiguous spans).

To inform the argument linking model about mention and type predictions made earlier in the pipeline, we incorporated those token-level predictions as one-hot features concatenated to the token embedding. We observed F1 scores for relational arguments of 34.8 (91.1% accuracy) and event arguments of 46.8 (93.7% accuracy). We prioritized recall over F1 score when choosing between similarly competitive models, since downstream precision can be improved post-linking (when producing a response). Using forced decoding of tags (by manually lowering the weight of the 'O' tag), we could artificially boost recall, but the tradeoff in precision under this method was observed in development to be too costly.

Given a hypothesis to condition on (Task 1b) we can use forced decoding selectively. By identifying which strings in the documents are similar to those in the hypotheses (described more in Section 4), we can force the tagger to decode non-O tags for those entities, thereby linking the hypothesized entities to an event or relation. Since this approach links entities that are hypothesized to be salient to the document, recall is improved. Since it does not link arbitrarily (which is the result of applying forced decoding uniformly), it does not result in a large drop in precision.

4 Name Similarity

For determining a match between a document mention and an entity string from a hypothesis, we employed a model for learning string similarity (Neculoiu et al., 2016). Given two strings, the model is capable of outputting a score (cosine similarity) that indicates how similar these two input strings are. The model is a Siamese network where two identical, parameter-shared modules are stacked upon the two input strings, with each string considered as a character sequence. Each module comprises of 4 bidirectional LSTM layers (size 64) followed by a feedforward layer that results in a vector of size 128. A cosine similarity function lays on top to connect these two Siamese branches. The model was trained using a dataset based on WikiNames, arising from earlier research at JHU (Andrews et al., 2012).

5 Knowledge Base Construction

Given sentence-level predictions of entities, relations, and events, we can construct a knowledge base for each document. This fundamentally relies on identifying coreferent entities, events, and relations in text. In this system, we only identify coreferent entities.

We used a state-of-the-art coreference resolution system (Lee et al., 2017) and ran it directly on the English text. We use a fuzzy overlap heuristic to resolve tokenization disputes between the coreference resolution system and the tokenization provided by LDC or the output translations.

The coreference resolution system produces clusters of spans, each of which we can identify with

[5] `nlp.stanford.edu/data/glove.6B.zip`

an entity. Due to memory constraints, large documents were broken into chunks of 5,000 tokens each and independently processed. In practice, this affected fewer than 5% of the documents. Furthermore, we could label each entity with a type determined by the majority predicted type among the mentions in its cluster. The confidence of the type is determined by the proportion of members of the cluster with that type.

As a result, typed document-level entities, not text spans, are the arguments of relations and events after this step.

6 Alignment

The use of state-of-the-art English entity, event, and relation extraction systems on translations of Russian and Ukrainian documents introduced two alignment problems. Firstly, discovered spans on the target side (English) needed to be aligned back to their source spans in order to reason about Russian and Ukrainian events and coreference chains. Secondly, queries given as character offsets in the source (Russian or Ukrainian) document needed to be aligned to a span discovered in the English translation in order to resolve the coreferences and relations. To address both problems, the document sentences and their respective translations were concatenated to the same bitext used to train the machine translation system and `fast-align` (Dyer et al., 2013) was run over the resulting augmented bitext, allowing alignments to be recovered.

Confirming the findings of Koehn and Knowles (2017) and Ghader and Monz (2017), we found that hard alignments obtained via neural attention were often incorrect, leading us to instead employ HMM-based alignment algorithms from the statistical machine translation literature. The choice to concatenate the documents to the training corpus rather than train on the bitext and decode the documents separately was motivated by concerns of domain mismatch between the bitext and the provided documents. Taking the union of the alignments in each direction (e.g. source to target, target to source) yielded pointers from English word tokens to source word tokens. Following these pointers from English to the source language and taking the maximum contiguous sub-span of the resulting sequence of source word indexes allowed English spans to be matched to their corresponding source tokens. Aligning query character offsets to spans in the translation presented a slightly more challenging problem since the span of translated word tokens to which the query tokens were aligned to needed to be a member of the set of spans discovered by the mention detection system – if this were not the case, coreference and event resolution would be impossible. To enforce this constraint, the corresponding span was chosen by computing the character index overlap between the span obtained by following the alignment pointers from the source and each of the spans discovered by the mention detection system (using F1 score as a metric), at which point the discovered span with the highest overlap score was chosen.

7 Query Handling

Output Knowledge Bases (KBs) under the evaluation were to be represented in the AIDA Interchange Format (AIF). We supported this via the USC ISI converter[6] which maps from TAC-KBP ColdStart format to AIF. We chose to use ColdStart as an intermediate representation because of its human-readability and general simplicity, which facilitated a faster development process.[7]

In order to evaluate the KBs during the pilot, NIST released three types of queries: 1) Class Queries, 2) Graph Queries, and 3) Zero-hop Queries in two formats: a) Simple XML, and b) SPARQL format, which participants had to execute themselves. Later in the evaluation NIST released a Docker container for applying the SPARQL queries directly on the AIF graphs. In the time available we were unable to make use of this service,[8] and also owing to comments from other performers on the slow speed of Jena when executing the SPARQL queries, we manually parsed and executed the Simple XML version of the queries. This ad-hoc Python script used the original TAC-KBP files as input, rather than the derived AIF format.

Class Queries: For each class query, for each KB, we iterated over all mentions of the entities of that type and serialized them to an xml file.

Graph Queries: A graph query is a tuple of (graph, entrypoints). A graph is a collection of edges, where each edge is a triple (subject, predicate, object). An entrypoint describes a node, and a typed descriptor, and we only processed two types of typed descriptors (string descriptor, and text descriptor). The string and text descriptor are analogous to string based search vs mention based search respectively. For the pilot evaluation, we had only one entry point entity per

[6] `https://github.com/NextCenturyCorporation/AIDA-Interchange-Format`

[7] While there is active development of utility libraries for AIF, these are Java-based, while our software stack was rooted in Python. The ColdStart-AIF converter was the path of least resistance for this initial pilot task.

[8] Out of the box attempts were unsuccessful, and on inspection the Perl-based software appeared to rely on an undocumented file hierarchy and naming structure for the AIF files; we left further exploration of this service for after the pilot.

graph query, though each entry point entity may have more than one descriptor. For each query, for each KB, we first searched for the entry-point entity by using either the string or the text descriptor of the entry point if that was applicable for the current KB. Then all edges where the resolved entry-point was either a subject or an object and for which we also had a predicate in the KB that matched the predicate in the query for that particular edge were marked as resolved edges and the edge with the highest confidence amongst those was chosen. This was we greedily built the returned graph starting from the entry-point as a resolved node, and then adding more nodes to the set of resolved nodes. Once the process could not be continued we stopped.

Zero-hop Queries: Zero-hop queries provide a mention of a filler or entity from a document (given as a character offset) and expect all references of that entity from the document as a result. In the simpler case of English documents, a query character offset is compared to the character offsets of all entities detected in the document, and the entity with the highest character overlap (as measured by F1 score between query and entity offsets) is chosen. All entities in the same coreference chain as the chosen entity are returned. Russian and Ukrainian documents present a slightly harder problem, as entity detection and coreference resolution for Russian and Ukrainian source documents was performed in their English translations, meaning that query offsets and their coreferent spans needed to be aligned from source to English and English to source, respectively. The alignment methods described in §6 yield pointers from Russian or Ukrainian words to their corresponding English translations. Using these alignment pointers, zero-hop responses for queries in Russian and Ukrainian were generated by computing the F1 overlap between the characters of the English words aligned to the source offset and each detected entity and subsequently following the pointers of all mention spans coreferent to the entity with maximum F1 back to the source.

References

Nicholas Andrews, Jason Eisner, and Mark Dredze. 2012. Name phylogeny: A generative model of string variation. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 344–355. Association for Computational Linguistics.

Chris Dyer, Victor Chahuneau, and Noah A Smith. 2013. A simple, fast, and effective reparameterization of ibm model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–648.

Hamidreza Ghader and Christof Monz. 2017. What does attention in neural machine translation pay attention to? *arXiv preprint arXiv:1710.03348*.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *Proc. ACL*.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. *arXiv preprint arXiv:1706.03872*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California. Association for Computational Linguistics.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 188–197, Copenhagen, Denmark. Association for Computational Linguistics.

Pierre Lison and Jörg Tiedemann. 2016. Opensubtitles2016: Extracting large parallel corpora from movie and TV subtitles. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation LREC 2016, Portorož, Slovenia, May 23-28, 2016.*

Paul Neculoiu, Maarten Versteegh, and Mihai Rotaru. 2016. Learning text similarity with siamese recurrent networks. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 148–157.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Sheng Zhang, Kevin Duh, and Benjamin Van Durme. 2018. Fine-grained entity typing through increased discourse context and adaptive classification thresholds. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, pages 173–179. Association for Computational Linguistics.