Adept Automatic Knowledge Discovery System for Cold Start Knowledge Base Population

Manaj Srivastava, David Trupiano, David Akodes, Ilana Heintz, Hannah Provenza, Bonan Min, Jay DeYoung, Lance Ramshaw, Roger Bock
Raytheon BBN Technologies
Cambridge, MA
{manaj.srivastava, david.trupiano, david.akodes, ilana.heintz, hannah.g.provenza, bonan.min, jay.deyoung, lance.ramshaw, roger.bock}@raytheon.com

Abstract

We submitted to the TAC 2017 Cold Start Knowledge Base Population (CSKB) track with our Adept Automatic Knowledge Discovery (A2KD) system. A2KD is an end-to-end knowledge base population system that uses diverse information-extraction (IE) algorithms provided by various universities, and integrates their output in a coherent way in order to populate a knowledge base. By the time of submission, the A2KD system had algorithms for Entity Discovery and Linking (EDL), Slot Filling (SF), and Event Argument Linking (EAL) for English and Chinese only. In addition to submitting system output for monolingual English and Chinese evaluations, we also submitted a bilingual variant which used English and Chinese but no Spanish.

1. Introduction

The A2KD system was developed under the DARPA-funded DEFT (Deep Exploration and Filtering of Text)[1] program. One objective of the DEFT program is to achieve population of a knowledge base by integrating information extraction algorithms from multiple providers. Using diverse algorithms from multiple sources helps ensure that A2KD uses the most reliable algorithm for any given kind of information (entities, entity-links, relations, events, etc.). A2KD provides a recipe for combining disparate information extraction algorithms into a single end-to-end system.

In this paper we describe the A2KD system and its variants that we used to submit the results to CSKB track. Section 2 provides a system overview. In Section 3, we briefly describe the various algorithms used for EDL, SF and EAL tasks for English and Chinese. In Section 4, we describe the mechanism employed by A2KD to integrate the output from different algorithms at the document level. Section 5 describes our techniques for doing a corpus-level integration of extracted information in a way that ensures construction of a coherent Knowledge Base (KB). Section 6 describes the final step of KB-level processing (KB Resolution). We identified certain discrepancies in the output KB that consistently resulted from a limitation in an A2KD module (e.g. chunk alignment), or in the output of one or more IE algorithms (e.g., Stanford's relation extraction would assign very high confidence values to some incorrect relations). This made us employ some post-processing steps that fix these discrepancies. In addition to that, A2KD also makes an attempt to do a more sophisticated cross-document event co-referencing. These steps make up the KB Resolver module of A2KD, and are described in Section 6. In Section 7, we describe the experiments we ran and the iterative improvements we made to the system.

[1] https://www.darpa.mil/program/deep-exploration-and-filtering-of-text

This research was developed with funding from the Defense Advanced Research Projects Agency (DARPA). The views, opinions and/or findings expressed are those of the author and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

Section 8 lists the variants of A2KD system that we used for our submission. In Section 9, we present the scores of our submissions, and conclude the paper in Section 10.

2. A2KD System Overview

A2KD system has the capability to populate a knowledge base from scratch, starting with raw text documents. Figure 1 outlines the architecture of the A2KD system.

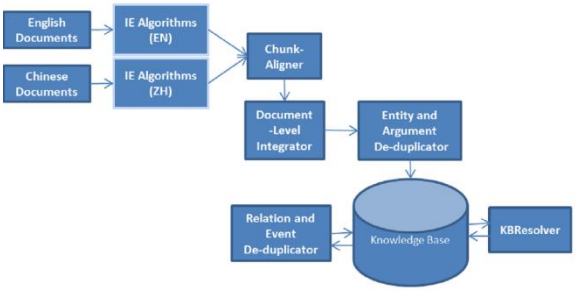

Figure 1 A2KD System Architecture

The core of the A2KD system includes modules implementing information-extraction algorithms for within document entity co-reference (entity coref), linking of entities into an external knowledge base (entity linking or Wikification), NIL-clustering of entities not linked to the knowledge base (NIL clustering), extraction of relations and their arguments (relation extraction), and extraction of events and their arguments (event extraction). A majority of these algorithms are implemented by various universities that participate in the DEFT program. The EAL algorithms for both English and Chinese are developed by BBN, so is the SF algorithm for Chinese. Table 1 gives a list of the algorithms used for English and Chinese languages along with the names of the providers.

A2KD reads the input text documents and passes them on to the various algorithm modules. Each algorithm module produces its own output in the form of an *HltContentContainer* object which abstracts a collection of entities (including entity-mentions and entity-links or NIL-cluster IDs), relations (including relation arguments), and events (including event arguments), along with their justifications in the source documents and confidence values assigned by the algorithms.

Once the algorithms are done processing a document, their output is combined in a coherent way. To this end, A2KD starts by doing chunk alignment. Chunk alignment is the process by which document-level entities extracted by the entity coref algorithm are co-referenced with the ones produced by EDL, SF, and EAL algorithms.

Algorithm Type	**Provider for English**	**Provider for Chinese**
Within Document Entity Co-reference	UIUC	Stanford
Entity Linking (without NIL-Clustering)	RPI	RPI
NILClustering[2]	UIUC, BBN	UIUC, BBN
Slot Filling	Stanford	BBN
Event Argument Linking	BBN	BBN

Table 1: List of Algorithm Providers for the A2KD System

A2KD uses this entity alignment information to create a document-level cross-algorithm entity co-reference mapping. This mapping is then used to attach to a coref entity any wikified or NIL-cluster IDs that the EDL algorithm has determined for the mapped EDL entity. Similarly, this mapping is used to replace an SF entity (or an EAL entity) with the mapped coref entity whenever the former is an argument of a relation (or event). This process of merging entity-level attributes or entity-replacement using document-level entity-alignment mapping is called document-level integration. Along with using entities extracted by the entity coref algorithm, A2KD also uses additional entities that are extracted by the entity-linking algorithm.

After the document-level integration, A2KD does a cross-document co-referencing of entities, non-entity arguments of relations and events, and the relations and events themselves. For entities, this co-referencing depends, among other things, on whether the entities are linked to the same ID in the external KB (or to the same NIL-cluster ID) as a result of entity linking (or NIL-clustering). For non-entity arguments, this co-referencing depends on the type and the textual span of the arguments. For relations and events, the co-referencing largely depends on the type of the relation or event, the roles of its arguments, and the particular entities or non-entity objects that form the arguments of that relation or event. Such a co-referencing also achieves a de-duplication of artifacts at the corpus level. Once the

[2] BBN's NIL-clustering is not part of the A2KD system, but was used as an alternative to UIUC's NIL-clustering in some of our submissions

entities and non-entity arguments have been de-duplicated, they are uploaded to the KB. This is done so that the subsequent relation and event de-duplication stages have access to non-duplicate arguments.

Thereafter, A2KD does a final step called KB Resolution which aims to fix certain kinds of discrepancies that may still exist in the KB. KB Resolution also does a more sophisticated cross-document co-referencing of events.

A2KD provides the end user the ability to search and view the content of the KB, or to dump the summary of the content to flat files using an in-built reporting module. For submission to TAC, we extended the reporting module to dump the entire content of the KB in the TAC specified format.

3. Algorithms Used in the A2KD System

In this section we will briefly describe the various algorithms used in A2KD system.

3.1 Within Document Entity Co-reference provided by UIUC (Illinois-Coref)

UIUC's DEFT Illinois Co-reference resolver is based on the Illinois-Coref system described in (Peng et. al. 2015), and uses models trained on the ACE 2004 dataset[3]. The Illinois-Coref system uses a machine learning approach to co-reference, with an inference procedure that supports straightforward inclusion of domain knowledge via constraints. The system first uses heuristics based on Named Entity Recognition, syntactic parsing, and shallow parsing to identify candidate mentions. A pairwise scorer generates compatibility scores for pairs of candidate mentions based on extracted features, subject to linguistic constraints. A left-to-right inference procedure then determines the optimal set of links to retain, incorporating constraints that may override the classifier prediction for a given mention pair. Illinois-Coref incorporates resources that allow detection of non-referring phrases and gender agreement between candidate mention pairs.

Illinois-Coref uses Illinois Named Entity Recognizer (INER, described (Redman et. al. 2016)) for Named Entity Recognition. During the integration of llinois-Coref in A2KD, we qualitatively determined that Illinois-Coref's entity-types were not very reliable. On the other hand, we found that the entity-types generated by INER were much better. We, therefore, decided to replace entity-types generated by Illinois-Coref with those generated by INER. However, we subsequently determined BBN's EAL algorithm and Stanford's SF algorithm to be the most reliable algorithms for entity-types. Therefore, in the final version of A2KD we decided to replace Illinois-Coref's entity-types with those coming from BBN's EAL algorithm, or Stanford's SF algorithm or INER, in that order.

[3] https://catalog.ldc.upenn.edu/LDC2005T09

3.2 Within Document Entity Co-reference provided by Stanford (StanfordCoref)

The Stanford algorithm for Chinese within document co-reference is described in (Lee et. al. 2013). It uses a sieve architecture that applies a battery of deterministic co-reference models one at a time from highest to lowest precision, where each model builds on the previous model's cluster output. The two stages of the sieve-based architecture, a mention detection stage that heavily favors recall, followed by co-reference sieves that are precision-oriented, offer a way to achieve both high precision and high recall. Further, this approach makes use of global information through an entity-centric model that encourages the sharing of features across all mentions that point to the same real-world entity.

3.3 Entity-Linking provided by RPI (RPI_EDL)

RPI's Entity-Linking algorithm utilizes a domain and language independent system (Wang et al., 2015), which is based on an unsupervised collective inference approach. Given a set of entity mentions M = {m_1, m_2, ..., m_n}, this system first constructs a graph for all entity mentions based on their co-occurrence within a paragraph. Then, for each entity mention m, it uses the surface form dictionary <f, e_1, e_2, ..., e_k>, where e_1, e_2, ..., e_k is the set of entities with surface form f according to their KB properties (e.g., labels, names, aliases), to locate a list of candidate entities $e \in E$ and compute the importance score by an entropy based approach (Zheng et al., 2014). Finally, it computes similarity scores for each entity mention and candidate entity pair <m, e> and selects the candidate with the highest score as the appropriate entity for linking. For Chinese, this system first translates mentions into English using name translation dictionaries mined from various approaches described in (Ji et al., 2009), then applies the same entity linking approach described above.

3.4 NIL-clustering provided by UIUC (Illinois Wikifier)

UIUC's DEFT Illinois Wikifier wraps the Cross-Lingual Wikifier described in (Ratinov et. al. 2011) and adds a NIL-clustering component. The Cross-Lingual Wikifier uses the inter-language links from Wikipedia[4] entries in 12 languages to train a set of cross-lingual embeddings based on words and Wikipedia titles in the target languages. Monolingual embeddings are computed for Wikipedia titles in each target language, replacing the titles to lexical contexts. Each language's monolingual embeddings are projected into a common space with English monolingual embeddings using alignment signals from the inter-language Wikipedia links. Dictionaries matching tokens and Wikipedia titles in each target language to Wikipedia titles in English are used to identify strings in input documents that may correspond to Wikipedia entries, and identify a set of candidate entries as the possible targets. A ranker computes the similarity of the context of the predicted mention with that of each candidate title to select the most likely title. The context is based both on the lexical embeddings of the predicted mention context, and a representation of predicted mentions and their targets in the mention context. A linear ranking SVM model is trained to combine these inputs, and the best-scoring candidate is used as the predicted title.

Some mentions will be identified as likely to refer to an entity, but have no viable target Wikipedia titles. The Nil Clustering component groups these unlinked mentions based on their character-level Jaccard similarity, using thresholds tuned on the NIST TAC 2016 Entity Discovery and Linking data set[5]. These thresholds set a minimum match on the character-level overlap between mentions and also the number of mentions that must surpass this threshold to allow formation of a group. The identifiers of such groups are set to the same value, indicating that they refer to the same unknown entity.

3.5 BBN's version of NIL-clustering (BBN_NilClustering)

BBN implemented a naïve NIL-clustering algorithm to cluster together similar entities that could not be assigned a wikified ID by RPI_EDL algorithm. For each of the entities without a wikified ID, a canonical mention is selected. The selected canonical mention is the NAM mention with the longest span. If a NAM mention is not available, we select the NOM mention with the longest span. Thereafter, these entities are merged based on exact match of the selected canonical mention's text. However, entities with differing types are not merged together.

3.6 Relation Extraction (SF) algorithm provided by Stanford (StanfordSF)

The algorithm uses a rule-based extractor along with a self-trained supervised extractor. The rule based extractor uses rules based on dependency graph structure, surface features, co-reference-chains, and edit-distance between an organization-name and a URL (to infer org:website relations). The self-trained supervised system is a logistic-regression based classifier with manually-crafted features and a Long Short Term Memory network classifier. This is further described in (Zhang et. al. 2016).

3.7 BBN's Relation Extraction algorithm (BBN_SF)

BBN's relation extraction or Slot Filling algorithm used for Chinese combines a set of Neural Network models and a pattern-based extractor. The algorithm is described in (Min et. al. 2017). The Neural Network models are two sets of Convolutional Neural Network (CNN) models, trained from ERE and an internally annotated dataset respectively. The pattern-based extractor applies proposition and lexical patterns to text to find relations. The Slot Filling component applies the CNN extractors and the pattern-based extractor sequentially, and takes a union of their results.

3.8 BBN's Event Extraction (EAL) algorithm (BBN_EAL)

For event extraction, we used BBN's system that was submitted to EAL track of last year's TAC evaluations. BBN's algorithm employs a simple form of joint inference, applies a variety of document-level inference rules, and a sieve-based event linking system to find events at the document level.

4. Chunk Alignment and Document-Level Integration

Chunk alignment is the process of co-referencing an entity from a certain "pivot" algorithm with entities from non-pivot algorithms by aligning the mention-spans of non-pivot entities to the mention-spans of pivot entities. This helps in ensuring that—despite

[4] https://www.wikipedia.org/

[5] http://nlp.cs.rpi.edu/kbp/2016/data.html

coming from unrelated source algorithms—the wikified or NIL-cluster ID for an entity, and the relations or events that it takes part in, all refer to the same entity.

For example, if from a certain document, the Illinois-Coref algorithm extracts a PER entity “Barack Obama”, while RPI_EDL extracts the same entity as “President Barack Obama”, chunk alignment ensures a mapping between “Barack Obama” and “President Barack Obama” so that the wikified ID of “President Barack Obama” can be assigned to the Illinois-Coref entity “Barack Obama”. Additionally, if StanfordSF extracts the same entity as “US President Barack Obama”, chunk alignment ensures that Illinois-Coref’s “Barack Obama” can replace StanfordSF’s “US President Barack Obama” in any relations the latter is an argument of. A2KD always treats the entity coref algorithm as the pivot algorithm.

We call this process of using entity alignments to merge relevant entity-level attributes or to replace entities from non-pivot algorithms with entities from the pivot algorithm document-level integration. Document-level integration essentially creates a composite data-structure (a composite *HltContentContainer* object) that contains output of all the algorithms in a way that appears to be the output of a single composite algorithm.

A2KD’s chunk alignment uses matching rules, as modified by algorithm-specific configurations, to determine if two textual spans (chunks) should be aligned to each other. The following matching rules are applied in disjunction with each other and in the order that they are listed in.

R1 (exact match): Two textual chunks should be aligned if their start and end offsets in the document’s text match, and the chunks represent the exact same string.

R2 (no prepositions match): Two textual chunks should be aligned if their end offsets match, and if the longer chunk does not contain any prepositions (e.g. “Barack Obama” should match with “President Barack Obama” but not with “daughter of Barack Obama”).

R3 (exact head match): If the two textual chunks have head spans, they should match if and only if the head-spans pass the R1 (exact match) test (e.g. “daughter of Barack Obama” should match “daughter” if both the spans for “daughter” have the same offsets in the document text).

In addition to the above rules, the following additional rules are applied if either text-span is an appositive mention. These rules were added to handle the mentions that come out of the BBN_EAL algorithm, which outputs appositives along with the modified noun-phrase, e.g. "US Senator from Massachusetts, Elizabeth Warren". This span would not match with either "US Senator from Massachusetts" (span-end not matching) or "Elizabeth Warren" (presence of a preposition).

R_APPO_1: If either mention is appositive and has a head span, we should align the two mentions in question only if the appositive’s head span matches with the other mention based on rule R2.

BBN_EAL uses the first child mention of an appositive as its head, so for the above example of appositive mention, the head from BBN_EAL would be "US Senator from Massachusetts", which would match with “US Senator from Massachusetts” coming from another algorithm. However, this would still not match if the mention coming from the other algorithm was “Elizabeth Warren”. To handle cases like those, we apply the following rule if rule R_APPO_1 has failed in finding the match.

R_APPO_2: If either mention is appositive and has a head span, and rule R_APPO_1 does not result in a match, the two mentions in question should match if the remaining span of the appositive mention (span without the head) matches with the other mention based on rule R2.

In addition to matching rules, chunk alignment also makes use of the following algorithm-specific configuration.

useRelaxedAlignmentRule: This configuration allows a chunk from an algorithm to align with a pivot chunk if it is contained in the pivot chunk, bypassing the other more specific rules. This kind of “relaxed” alignment was added to address the issue of some Illinois-Coref entities with very long spans missing any alignment with StanfordSF or BBN_EAL entities at all. The types of mentions to be aligned, and also the types of the entities that the mentions map to, must nonetheless match in order to weed out noisy alignments. Currently, this configuration is used only for StanfordSF and BBN_EAL entities when the entity coref algorithm is Illinois-Coref and the language is English.

5. Cross-Document De-duplication

After integrating artifacts from various algorithms at the document level, A2KD does a merging or de-duplication of these artifacts at the corpus level. This is done in order to ensure uniqueness of these artifacts. For example, if there are entities from two or more different documents that link to the same wikified ID, they are essentially a single real world entity, and are therefore merged by A2KD to create a single Entity object. Similarly, if there's a per:resident relation for that entity that is extracted from two or more documents, all instances of that relation need to be merged to ensure that A2KD has justifications for the same real world relation from all the documents that relation appeared in. Note that merging of events in this stage is only partial, in the sense that the merged events are required to have the exact same set of entities or non-entity objects as arguments. So, while two contact.meet events with the same entity and date arguments will be merged together, they will not be merged with a contact.meet event with the same entity and date arguments but an additional place argument. Such merging happens in the KB Resolver stage.

When merging artifacts, A2KD makes sure that the final merged or de-duplicated artifact contains all the relevant information from artifacts contributing in the merging. This includes taking a union of all the provenances. In this section, we will describe what forms the basis to merge artifacts of different types (entities, non-entity arguments, relations, and events) and how various attributes (like confidence and provenances) for those artifacts are merged.

Table 2 gives a list of attributes for different artifact types that A2KD uses to determine the uniqueness of artifact of that type.

Merging of the artifacts entails merging their attributes. The attributes determining the uniqueness of an artifact do not need any merging since they are shared across all the artifacts to be merged. Other attributes do require merging. Merging of provenances or justifications for any artifact type is done by taking a union of provenances or justifications attached to all the contributing artifacts that are taking part in the merging. So, for entities, the merging of provenances would entail taking a union of all the entity-mentions for the contributing entities. For relations, merging of provenances would entail taking a union of justifications from all the documents that the contributing relations have appeared in, and so on.

Merging of the artifacts entails merging their attributes. The attributes determining the uniqueness of an artifact do not need any merging since they are shared across all the artifacts to be merged. Other attributes do require merging. Merging of provenances or justifications for any artifact type is done by taking a union of provenances or justifications attached to all the contributing artifacts that are taking part in the merging. So, for entities, the merging of provenances would entail taking a union of all the entity-mentions for the contributing entities. For relations, merging of provenances would entail taking a union of justifications from all the documents that the contributing relations have appeared in, and so on.

Artifact Type	**Attributes determining the uniqueness of the artifact**
Entity	Wikified or NIL-cluster ID + Entity Type
Non-Entity Text Chunk (e.g. Date, Title, Number, Temporal Value etc.)	Value of the text chunk + Type of the artifact (if the artifact has a type)
Relation or Event Argument	Argument Role + Uniqueness attributes depending on the artifact-type of the argument
Relation	Relation Type + Uniqueness attributes of all the arguments
Event	Event Type + Uniqueness attributes of all the arguments

Table 2 List of attributes determining uniqueness of artifacts

Merging of the artifacts entails merging their attributes. The attributes determining the uniqueness of an artifact do not need any merging since they are shared across all the artifacts to be merged. Other attributes do require merging. Merging of provenances or justifications for any artifact type is done by taking a union of provenances or justifications attached to all the contributing artifacts that are taking part in the merging. So, for entities, the merging of provenances would entail taking a union of all the entity-mentions for the contributing entities. For relations, merging of provenances would entail taking a union of justifications from all the documents that the contributing relations have appeared in, and so on.

The following subsections describe how other attributes are merged.

5.1 Merging Entities

Merging Canonical Mentions

The A2KD system stores a KB-wide canonical mention for each entity, for use in the A2KD UI when viewing information for an entity. It is possible for entities with different canonical mentions to have the same wikified or NIL-cluster ID. For example, the canonical mention for an entity can be "President Obama" in one document, "Barack Obama" in another document, and "POTUS 44" in yet another document. These canonical mentions can also have different confidences. For example, the entity coref algorithm can be 100% confident that "President Obama" is the correct canonical mention for the entity it extracted from one document, but only 95% confident that "POTUS 44" is the canonical mention of an entity extracted from another document.

The value of the merged canonical mention is determined as the most frequent value of the most confident canonical mentions of the entities to be merged. The confidence of the merged canonical mention is the same as the confidence of the merged entity (see below).

Merging Entity Confidences

The confidence of the merged entity is a weighted average of confidence of individual entities, weighted by number of entity mentions linked to each entity.

5.2 Merging Relations

Merging Relation Confidences

The confidence of a merged relation is a weighted average of confidence of the individual relations, weighted by the number of provenances of each relation.

Merging Relation Argument Confidences

The confidence of a merged argument is a weighted average of the confidence of that argument in the contributing relations, weighted by the number of provenances of that argument in the contributing relations. If the number of provenances is zero for any argument of the relation, the weight used for the weighted average is the number of provenances of parent relation.

5.3 Merging Events

Merging Event Confidences

The confidence of a merged event is a weighted average of the confidence of the individual events, weighted by the number of provenances of each event. If an event does not have a confidence (BBN_EAL produces events without a confidence value), its confidence value is taken as the minimum of the confidence values of its arguments. If none of the arguments has a confidence value either, a default value of 0.5 is used.

Merging Argument Confidences

The confidence of a merged argument is a weighted average of the confidence of that argument in the contributing events, weighted by the number of provenances of that argument in contributing events. If the number of provenances is zero for any argument of the event, the weight used for the weighted average is the number of provenances of parent event. If the confidence value for an argument is not available, the confidence of the parent event is used. If the confidence of the parent event is not available either, a default value of 0.5 is used.

6. KB-Level Processing (KB Resolution)

In the KB produced by above mechanism, we could still find certain inconsistencies. For example, we often found that more than one entity—with differing types—were assigned the same wikified (or NIL-cluster) ID. Sometimes this could be attributed to limitations in entity-typing of the IE algorithms, while other times this was due to a limitation of the chunk alignment algorithm. Similarly, we saw opportunities to improve the output of certain algorithms in ways that would make more sense, or to make the content of the KB more presentable to an end-user. For example, we found that StanfordSF can sometimes assign very high confidence values to many incorrect relations. In order to fix such issues, we run a post-processing step in A2KD which we call KB Resolution (since it attempts to resolve the KB to a more coherent state).

Resolving Multiple Entities with the same Wikified ID

A qualitative analysis of the cases where multiple entities were linked to the same wikified or NIL-cluster ID showed that the linked entities usually differed in their type (e.g. JFK (LOC) and John F. Kennedy (PER) would both be linked to the wikified ID for John F. Kennedy (PER)). The correct entity out of the ones linked would usually be the entity

with the most mentions. We therefore resolved this issue by keeping the entity with the most entity mentions, and removing all other entities from the KB. In order to ensure overall consistency, we also removed any relations or events that contained the removed entities as an argument. While we did not do a quantitative impact analysis on SF or EAL scores for any fixes made in KB Resolution stage, we did manually review sample output. A qualitative inspection of the entities (with their corresponding relations and events) removed showed that most deletions were appropriate.

Resolving Confidences for StanfordSF Relations

We scale down the confidence of all relations to 70% of the StanfordSF-assigned confidence. We experimented with raising the confidence of relations with many justifications, and with further lowering the confidence of relations for which high-frequency arguments were not often linked, but these changes had little effect.

Resolving Entities with Incorrect Mentions

Due to limitations of our chunk alignment algorithm or the output of coref or entity linking algorithms, it is possible for a merged entity to have some incorrect mentions that actually belong to a different entity. For example, when both George H.W. Bush and George W. Bush appear in a document, it is possible for the coref algorithm to mix up a mention of “Bush” for George H.W. Bush with a mention of “Bush” for George W. Bush. If the mentions for either entity are too few and the confidences assigned to them are nearly same, the mixed up mention can result in the chunk alignment algorithm aligning George W. Bush from coref algorithm with George H. W. Bush from entity linking algorithm, and merging their mentions. For the final entity for George W. Bush, it is possible for there to be a small set of mentions with canonical string George H.W. Bush depending on how many documents the two entities appeared in and were co-referenced incorrectly in.

To address such cases, for entities with a very high number of mentions, we look for mentions that have canonical strings that represent fewer than 15% of all of the mentions. These mentions are split into separate entities. Relations and events involving these entity mentions as arguments are also edited to point to the correct entity. Note that the split entity would still have the same wikified (or NIL-cluster) ID as it had originally. Since, this step of entity-splitting is done after the step of dropping entities with minority type that have the same wikified (or NIL-cluster) ID (as explained above), we do not end up losing the split entities.

Cross-Document Event Co-reference

The event co-referencing that happens earlier in A2KD (as described in Section 5.3) merges events that have the exact same set of arguments. This may be insufficient in cases where the same real world event appears with different sets of arguments in different documents. To address this issue, in the KB Resolution stage, we take an additional pass at cross-document event co-referencing.

We use an implementation of the cross document event co-reference system used by BBN in the 2016 TAC Event Arguments evaluation. It uses high precision rules based on events, arguments, roles, and cross document entity co-reference or EDL to determine which events to link. An event is a collection of event arguments of identical event type; an event argument consists of a filler (entity, a phrase such as "twelve monkeys" or "life in prison", or a time), a role, an event type, and a probability distribution over realis values. The overall function of the cross document event co-reference algorithm can be thought of as a process of filling in the third partition of a tripartite graph. This graph consists of partition (A), the event arguments; partition (B), the events themselves; and partition (C) the cross document events, where an edge between (A) and (B) indicates a membership relation of an argument to an event, and an edge between (B) and (C) represents an equivalence relation between events.

(A) and (B), and all edges between them come from the EAL output of the A2KD KB. We discard any fillers from (A) that have an entirely Generic realis, and any roles not used in any of our rules. The edges between (B) (events) and (C) (cross document events) are built in two steps: (1) by joining any pair of events that share at least two arguments in the

same role/filler combination (for a limited set of roles), have no more than a total of eight fillers between the two events, and have no more than four fillers in any resulting role; (2) by a reification process described below.

For efficiency purposes, step (1) of forming the cross document events is done via a partition by event type, and a partition by shared event arguments. A limitation of this design is that the edges initially output by (1) are not equivalence relations, because each event will appear in a different partition (or multiple partitions) and be processed separately, thus the initial links from (B) to (C) are not proper equivalence relations. To make them equivalence relations, we reify the output so for any three events (x), (y) and (z), where (x), (y) are in (C_1) and (y), (z) are in (C_2), (C_1) and (C_2) become the same partition. This is done via a simple connected components algorithm using a breadth-first search across partitions (B) and (C). Finally, we form confidence and realis values for the output events by taking an average weighted by the amount of justification each has.

7. Experiments and Results

Dataset Used

For our experiments to evaluate the A2KD system, we used the datasets used for TAC evaluations in 2016[6].

The dataset used for our English EDL experiments was a set of 169 documents, out of which 84 were newswire documents and the remaining were discussion forum documents. For our English SF experiments, the dataset consisted of a total of 5387 documents, 3836 out of which were newswire documents, and the remaining were discussion forum documents.

Given the constraints of time, we were not able to do experiments or scoring with A2KD for EAL, or for Chinese EDL or SF.

Experiments Run

Most of our experiments dealt with improving the EDL scores for English. This was chosen as the main focus because the SF and EAL scores depended on the quality of EDL output. Below we describe our EDL and SF experiments for English.

English EDL Experiments

Most of the English experiments were run without KB Resolver. This was because with our initial experiments, we found KB Resolver to have only marginal effect on the EDL scores, and also because we identified more scope of improvement in the core A2KD modules of chunk alignment and document-level integration.

Most of our experiments also did not include UIUC's NIL-clustering, since it became available a little later. Also, note that for EDL experiments, the chunk alignment configuration useRelaxedAlignmentRule (as described in section 4) was not used. This rule was added during SF score optimization as will be explained in the following sub-section. Nonetheless, adding this rule did not have any effect on the final best EDL scores that we obtained with our EDL experiments.

For our EDL experiments, we focused only on the strong_typed_mention_match metric, and all the scores we report here pertain to that. These scores are summarized in Table 3. Note that for some of the experiments, we failed to record the P or R values; these are marked as NA in the rows corresponding to those experiments.

Our baseline EDL F-score was 0.426. This was using mention-heads from BBN_EAL and entity-types from StanfordSF and INER (in that order). Subsequently, we decided to prefer entity-types from BBN_EAL over StanfordSF, since we qualitatively observed BBN_EAL entity-types to be the most reliable of all IE algorithms used. Since the quality of entity-type reassignment depends on the quality of chunk alignment, we made certain improvements to the latter. We added the alignment rules R_APPO_1 and R_APPO_2 (as described in Section 5) to better align appositive mentions. We also found that the A2KD system was not providing the text from HEADLINE element of newswire documents to the algorithm-modules. This was resulting in missing any mentions from the headline text. After fixing this, and including a couple other small fixes like dropping the leading determiner from a mention-span, and dropping certain mentions that were group words (like people, group, crowd, etc.), the EDL F-score showed significant improvement and jumped to 0.512.

[6] http://nlp.cs.rpi.edu/kbp/2016/data.html

Neither Illinois-Coref nor RPI_EDL extracts names of posters from discussion forum (DF) documents as mentions. Each DF post has the poster name mentioned in its metadata. This made us augment the A2KD KB with discussion forum poster names as a post-processing step. By doing that, we saw an increase of nearly 6.6% in the EDL F score.

On this version of A2KD, we ran KB Resolver, and got a small improvement of 0.3% in F-score. Note that none of the experiments reported until this point had KB Resolver run on the final KB. From this point onwards, all improvements were done as post-processing steps augmenting the KB produced by KB Resolver.

Similar to adding poster names from DF documents, we also added text from AUTHOR elements of newswire documents as mentions. This implementation had a small bug, however, which extracted the text from author elements as is, even when it had author names followed by their affiliation (organization names). We fixed this bug before our final submissions, but not for this experiment, which got us a small increase in F-score of 0.2%.

We then realized that our A2KD output also included plural nominal mentions, which the TAC reference files did not expect. Dropping plural nominals brought the F-score to 0.630.

Features/Improvements	P	R	F
Baseline (with BBN_EAL mention heads)	0.406	0.449	0.420
BBN_EAL types + better chunk alignment+headline+group-word fix	0.456	0.583	0.512
Using DF poster names	0.499	0.687	0.578
Above improvements with KB Resolver	0.505	0.683	0.581
Using newswire author names	NA	NA	0.583
Dropping plural nominals	0.583	0.686	0.630
Dropping long mentions	NA	NA	0.651
Changing type of "government" from NAM to NOM	NA	NA	0.674

Table 3 Summary of English EDL Experiments

We found that some of the mentions were too long. These were mostly bogus mentions that started with a URL. We also found that filtering out nominal mentions with more than a certain number of tokens helped improve the scores. We experimented with dropping nominal mentions that were longer than 2 tokens or 1 token, and found that we got the best scores by dropping nominal mentions longer than a single token. By dropping bogus mentions and nominal mentions longer than one token, we could improve the F-score to 0.651.

As a final post-processing step, after finding that numerous instances of the mention "government" were tagged with type NAM instead of NOM, we made a specific fix for these cases to change the mention type to NOM. This gave us an improvement of 2.3% in F-score.

After the above improvements to the EDL system, we shifted our focus toward improving the SF scores.

English SF experiments

For our SF experiments, we report only the 0-hop micro P, R and F scores. A summary of the iterative improvements can be found in Table 4 (rounded to 3 significant digits).

Our baseline F-score was 0.1026. We found that we were losing a good number of relations from StanfordSF because many of its entities were not getting aligned with Illinois-Coref entities. This was resulting in dropping of any relations that had these unaligned entities as arguments. On analyzing this issue, we found that the failure of alignment was due to some very long mention-spans coming out of Illinois-Coref which were failing alignment with shorter yet reasonable StanfordSF mention spans given our existing set of chunk alignment rules. This made us add the useRelaxedAlignmentRule rule for chunk alignment (as described in section 4). After adding this rule, our SF F-score moved up by a little over 1% to 0.1143.

Features/Improvements	P	R	F
Baseline	0.356	0.060	0.103
Relaxed chunk alignment	0.313	0.070	0.114
Using naïve NIL-clustering	0.335	0.071	0.117
Fixes in A2KD and StanfordSF	0.379	0.090	0.145
Using UIUC NIL-clustering	0.373	0.090	0.145

Table 4 Summary of English SF Experiments

The other main reason for low F-score was found to be absence of corpus-level NIL-clustering. Since

UIUC's NIL-clustering was still not a part of A2KD, we used BBN's naïve NIL-clustering implementation instead. With naïve NIL-clustering, the scores improved a tiny bit to 0.1174. We subsequently fixed a few bugs in A2KD and updated StanfordSF with a version that produced provenances for relation arguments. Additionally, we made a fix in A2KD to reassign entity-types for entities that filled an SF slot based on the expected type for that slot. For example, if an LOC entity appeared in the "Victim" slot of a Life.Die relation, A2KD would reassign the type of that entity to PER. These changes brought the score to 0.1453. Finally, we ran an experiment with all the above fixes but using UIUC NIL-clustering instead of BBN's naïve NIL-clustering and the final best F-score that we obtained was 0.1449.

8. Description of Systems Submitted for Evaluation

We submitted output KBs from 10 different variants of A2KD system for evaluations: 4 for Chinese, 4 for English and 2 for cross-lingual evaluation.

The variants of Chinese systems differed in the NIL-clustering algorithm employed (UIUC or BBN), and whether the BBN SF algorithm was optimized for a higher precision or a higher recall. These systems were as follows:

A2KD_Adept_KB_CMN_1 (CMN_1): This system used UIUC's NIL-clustering and BBN SF component optimized for higher precision.

A2KD_Adept_KB_CMN_2 (CMN_2): This system used UIUC's NIL-clustering and BBN SF component optimized for higher recall.

A2KD_Adept_KB_CMN_3 (CMN_3): This system used BBN's naïve NIL-clustering and BBN SF component optimized for higher precision.

A2KD_Adept_KB_CMN_4 (CMN_4): This system used BBN's naïve NIL-clustering and BBN SF component optimized for higher recall.

Similar to Chinese variants, the variants of English systems also differed in the NIL-clustering algorithm employed. In addition, these variants differed in whether a fix for per:title relations ("pertitle fix") was added on top of the output of the Stanford SF algorithm. This fix would look for per:title relations where the same title was attributed to more than one person entities in the same sentence, and would resolve the attribution in favor of the person entity whose textual span is closer to the span of the extracted title. The English variants were as follows:

A2KD_Adept_KB_ENG_1 (ENG_1): This system used UIUC's NIL-clustering and had the pertitle fix.

A2KD_Adept_KB_ENG_2 (ENG_2): This system used UIUC's NIL-clustering and did not have the pertitle fix.

A2KD_Adept_KB_ENG_3 (ENG_3): This system used naïve NIL-clustering and had the pertitle fix.

A2KD_Adept_KB_ENG_4 (ENG_4): This system used naïve NIL-clustering and did not have the pertitle fix.

For the cross-lingual variants, the only NIL-clustering algorithm used was BBN's naïve NIL-clustering, since UIUC's NIL-clustering does not support NIL-clustering across different languages. These systems, therefore, differed only based on whether the BBN Chinse SF algorithm used was optimized for higher recall or higher precision.

A2KD_Adept_KB_XLING_1 (XLING_1): This system used the BBN SF component optimized for higher precision.

A2KD_Adept_KB_XLING_2 (XLING_2): This system used the BBN SF component optimized for higher recall.

9. Results

In this section, we present the scores of our systems.

Since none of our submissions included Spanish, we are not presenting the scores of any cross-lingual evaluations, since those would be artificially low.

Our scores for monolingual EDL evaluations for English and Chinese are listed in Table 5 (to avoid repetition, we do not report scores from systems that only differ in SF algorithm). We are only reporting the scores for strong_typed_mention_match (stmm), strong_typed_all_match (stam), strong_typed_link_match (stlm), and strong_typed_nil_match (stnm) metrics.

For monolingual SF evaluations, we report K3 0-hop Average Precision (AP) scores for SF-ALL-Macro (sfam-0) and K3 ALL-hop AP scores for LDC-MEAN-ALL-Macro (lmam-ALL) metrics. We report these scores for only our best-performing systems (based on SF-ALL-Macro scores). For the ALL-slots,

EVENT-slots-only and SF-slots-only evaluations, the scores for both English and Chinese are listed in Table 6. For both English and Chinese, our systems with BBN's naïve NIL-clustering (ENG_3, CMN_4) did better than other systems. For the reported metrics, ENG_4 did as good as ENG_3. The recall-optimized algorithm for Chinese (CMN_3) also seems to have helped in outperforming other variants.

Language	System Name	stmm F	stam F	stlm F	stnm F
English	ENG_1	**0.689**	**0.493**	**0.518**	**0.461**
	ENG_3	0.688	0.489	0.513	0.459
	XLI_1	0.570	0.431	0.500	0.342
Chinese	CMN_1	0.608	0.408	0.440	0.332
	CMN_3	0.606	0.408	0.439	**0.335**
	XLI_1	**0.611**	**0.410**	**0.443**	0.332

Table 5 System-wise EDL scores

Language	Slot-type	System Name	sfam-0 AP	lmam-ALL AP
English	ALL	ENG_3	0.0954	0.0963
	EVENTS	ENG_3	0.1490	0.1644
	SF	ENG_3	0.0907	0.0807
Chinese	ALL	CMN_4	0.0918	0.0899
	EVENTS	CMN_4	0.1099	0.1082
	SF	CMN_4	0.1141	0.1102

Table 6 Best scores for K3 SF queries

10. Conclusion

In this paper, we have described our submissions to the TAC CSKB track. Our system, A2KD, is a recipe for combining the multitude of information extracted by various open source information extraction algorithms in order to populate a Knowledge Base. We discussed the mechanism employed by A2KD to ensure that the document-level information extracted by the contributing algorithms is coherently integrated at the corpus-level. The techniques used to that end included 1) chunk alignment and document-level integration, and 2) corpus-level co-referencing. The KB Resolution stage then uses information available at the corpus-level to further refine the quality of the assertions stored in the KB.

We also discussed the experiments we did to improve the accuracy of our final A2KD output, and our results from this year's TAC evaluation.

11. References

Haoruo Peng, Kai-Wei Chang Dan Roth, A Joint Framework for Coreference Resolution and Mention Head Detection, CoNLL (2015) pp.10

Tom Redman and Mark Sammons and Dan Roth, Illinois Named Entity Recognizer: Addendum to Ratinov and Roth '09 reporting improved results, Tech Reports (2016)

Heeyoung Lee, Angel Chang, Yves Peirsman, Nathanael Chambers, Mihai Surdeanu and Dan Jurafsky, Deterministic Coreference Resolution Based on Entity-Centric, Precision-Ranked Rules, Computational Linguistics (December 2013) p.885-916

Han Wang, Jin Guang Zheng, Xiaogang Ma, Peter Fox, and Heng Ji, Language and domain independent entity linking with quantified collective validation, EMNLP (2015)

Jin Guang Zheng, Daniel Howsmon, Boliang Zhang, Juergen Hahn, Deborah McGuinness, James Hendler, and Heng Ji, Entity linking for biomedical literature, BMC Medical Informatics and Decision Making (2014)

H. Ji, R. Grishman, D. Freitag, M. Blume, J. Wang, S. Khadivi, R. Zens, and H. Ney, Name extraction and translation for distillation, Handbook of Natural Language, Processing and Machine Translation: DARPA Global Autonomous Language Exploitation (2009)

L. Ratinov and D. Roth and D. Downey and M. Anderson, Local and Global Algorithms for Disambiguation to Wikipedia, ACL (2011)

Yuhao Zhang, Arun Chaganty, Ashwin Paranjape, Danqi Chen, Jason Bolton, Peng Qi, Christopher D. Manning, Sealing Pipeline Leaks and Understanding Chinese, TAC KBP (2016)

Bonan Min, Zhuolin Jiang, Marjorie Freedman and Ralph Weischedel, Learning Transferable Representation for Bilingual Relation Extraction via Convolutional Neural Networks, To appear in IJCNLP (2017)