CRIM's Systems for the Tri-lingual Entity Detection and Linking Task

Gabriel Bernier-Colborne, Caroline Barrière, Pierre André Ménard
Computer Research Institute of Montréal
405, Ogilvy Avenue, suite 101
Montréal (Québec), Canada, H3N 1M3
g.b.colborne@gmail.com, caroline_barriere@yahoo.ca,
menardpa@crim.ca

Abstract

This report describes the system developed by the CRIM team for the tri-lingual entity detection and linking (TEDL) task in the knowledge base population (KBP) track at TAC 2017. The entity mention extraction module exploits a deep neural network to extract features from texts, and either a CRF tagger or n-gram classifier to detect entity mentions. Transfer learning is used to exploit training data in which the set of classes is different than those in this year's evaluation data. The linking module uses a variety of linguistic and statistical methods to link mentions to the reference knowledge base. For our first participation in the TEDL task, we focused on a single language, English, and submitted four runs in order to compare the two classifiers and assess the impact of the transfer learning technique.

1 Introduction

The goal of the tri-lingual entity detection and linking (TEDL) task is to automatically detect both named (NAM) and nominal (NOM) mentions of five types of entities in a collection of texts: people (PER), organizations (ORG), geopolitical entities (GPE), locations (LOC), and facilities (FAC). The mentions must then be linked to a reference knowledge base (KB). Mentions referring to entities that are not encoded in the KB must be marked as *NIL*, and NIL mentions referring to the same entity must be clustered together across documents.

The task is basically the same as last year's edition of the TEDL task, therefore last year's evaluation data is a valuable source of training data. Data from previous editions of the task could also be used for this purpose, but it is important to note that the set of mention types and entity types has changed as the task evolved. For instance, in the 2015 training and evaluation data, NOM mentions were only annotated for PER entities. In this work, we experimented with transfer learning in order to use these datasets to train the entity mention extraction module, as explained in the following section.

For our first participation in the TEDL task, we decided to focus on English, and to explore both deep learning methods and more ad hoc methods based on corpus statistics and linguistic heuristics. Our objective was not only to build an efficient and accurate system, but also to challenge and explore various ideas, regardless of their trendiness.

2 System description

The system we developed has two main components: an entity mention extraction (EME) module, and a coreference resolution and linking (CRL) module. These two components are described in the following subsections.

2.1 Entity mention extraction

Our approach to detecting entity mentions exploits machine learning almost exclusively, aside from minimal preprocessing, as well as one post-processing step used to extract discussion forum post authors, using simple regular expression matching.

All other entity mentions are extracted using a deep neural network trained on the labeled datasets used in past editions of the TEDL task (LDC2017E03). For 2 of our submissions, we train the model on the 2016 evaluation dataset only. For the 2 others, we train the model on both the 2015 and 2016 datasets. Since the set of labels changed from one year to the next, we opted for a transfer learning approach, whereby a first model is trained on the 2015 data, then a second model is trained on the 2016 data using the feature extractor which was previously trained on the 2015 data. The 2016 model is trained in two stages: first we train only the classifier (output layers), then we fine-tune the whole model.

The feature extractor employs two recurrent neural networks (specifically, LSTMs): one at character level (to represent the morphology or spelling of words, as well as case information) and one at word level (to represent words and their context). For each word, the final hidden state of the character-level LSTM is concatenated to a pre-trained word embedding and fed to a second LSTM which outputs a feature vector for each word. These features represent the spelling, meaning and context of each word. In the output layers of the model, a classifier or tagger takes the features representing the words in a sentence and predicts the span and type of the entity mentions in the sentence.

To predict the entity mentions based on the features obtained for each word, we tested 2 types of classifiers: a CRF tagger and a combination of n-gram classifiers. These are described below. In both cases, the model uses only the sentence as context. The hidden state of the (word-level) LSTM is re-initialized for each input sentence, thus it does not take into account past sentences.

2.1.1 CRF

The first classifier we tested is a conditional random field (CRF), which tags words sequentially, based on the features representing those words and their context. CRF taggers have long been used for named entity recognition, traditionally using a variety of manually defined features (cf. Sil et al. (2016)) rather than the output of a neural feature extractor. The linear-chain CRF we use considers not only a word's features when predicting its tag, but also the previous tag. We use IOB tags, and concatenate the mention type and entity type of each mention to the B tag corresponding to the first word in the mention (e.g. B+PER+NOM), which gives us a total of 12 possible tags. The loss function we use for training is the difference, for a given sentence, between the score of the gold tags and the score of the predicted tags. This score is computed by summing the log-probabilities of each tag given the features and the previous tag. The forward algorithm is used to compute the partition function, and the Viterbi algorithm is used for decoding.

2.1.2 N-gram classifiers

To allow for nested mentions (e.g. [[Yorkshire] Ripper]), we also tested a combination of n-gram classifiers trained jointly on the labeled sentences. This approach to classification is somewhat similar to that of the YorkNRM team at last year's edition of the TEDL task (Xu et al., 2016), in that we locally classify fragments of different lengths, though our approach to feature extraction is completely different.

Given a sentence, for each value of n from 1 to some threshold t, the classifier for that value of n is used to classify all n-grams, using the cartesian product of the mention types and entity types as classes, as well as the negative class, for a total of 11 classes. Each classifier is a simple softmax classifier. The loss used for training is the average, for each value of n, of the average cross-entropy of all the n-grams in a sentence for that value of n. We set the threshold $t = 5$, as very few mentions in the 2016 gold standard were longer than this (about 0.5% by our count, using the simple tokenizer used in the EME module). Unlike Xu et al. (2016), we did not sub-sample negative examples or use heuristics to avoid overlapping mentions.

2.1.3 Training details

We used PyTorch[1] to train the neural networks. For feature extraction, we used one bi-directional LSTM layer (with 50 units) at character level, and two bi-directional LSTM layers (with 200 units each) at word level, with dropout between the two layers (with dropout probability 0.25). The Adam optimizer (Kingma and Ba, 2014) with $\beta_1 = \beta_2 = 0.9$ was used for parameter optimization. We kept a small validation set to do early stopping (after 10 checkpoints with no improvement on the validation set, with one checkpoint after every batch of 25 training documents and a maximum of 10 epochs over the training set). Gradient clipping was also used for regularization.

We used pre-trained, 200-dimensional GloVe word embeddings[2] trained on Wikipedia and Gigaword. We added a random embedding for unknown words. The embedding lookup function looked up the lower-case form of the word if it

[1] http://pytorch.org

[2] https://nlp.stanford.edu/projects/glove/

was not found as is. The word embeddings were not fine-tuned during training.

Pre-processing was limited to character normalization[3], sentence splitting[4] and tokenization[5].

2.2 Coreference resolution and linking

In this section, we describe the second module of our system, which we call the coreference resolution and linking (CRL) module as the construction of coreference chains (CCs) is central to the linking strategy.

For a given document, the CRL module receives a set of mentions (also called queries) from the EME module described in the previous section. These queries have already been assigned an entity type (i.e. PER, LOC, GPE, FAC or ORG) as well as a mention type, which is either NAM (for named entities) or NOM (for nominal mentions). The goal of the CRL module is to discover the set of entities referred to in the text, and link the various mentions to these entities.

Before we explain in detail the various components of the CRL module, let us describe the underlying principles that guided its development.

A first assumption is that most entities will be explicitly referred to by one or more NAM mentions in the text, which can have coreferring NOM mentions. For example, in a text about events happening in Egypt, a NAM mention "Morsi" would likely be used to provide an explicit reference to Egypt's president [Mohammed Morsi][6], and any number of NOM mentions (e.g. "man", "president") could also refer to this entity.

Only in rare cases will an entity not be referred to by a NAM mention and instead be implicit (understood from context), in which case a set of coreferring NOM mentions would point to this entity. For example, in one news article, the text would implicitly talk about [Grand Ethiopian Renaissance Dam] using a series of NOM mentions (e.g. "dam"), but no NAM mentions.

A second, related assumption is the so-called "one-sense-per-discourse" hypothesis. We assume that identical mentions (e.g. "Obama") in the text are all linked to a single entity (e.g. [Barack Obama]) regardless of where they occur in the text. This creates an obvious limitation as our system will not be able to link ambiguous mentions to different entities within the same text. This limitation might become problematic if the texts were longer, but is perhaps an acceptable simplification when dealing with short texts.

A third assumption is that each text to be analyzed (news article or discussion forum) will have some topical coherence, that is we assume a news article will carry a coherent message about a particular event or topic, and likewise a discussion forum will be focused on one topic. We do not assume that all entities in a given text will form a single, coherent set, but we do assume that texts will contain coherent subsets of entities.

A fourth assumption is that the documents to be analyzed are *real* documents, written by people, and may therefore contain typographical errors.

A fifth assumption is that we can use Wikipedia as a proxy for BaseKB, given the fact that a large number of nodes in BaseKB have a corresponding Wikipedia page. We will therefore perform candidate generation using pre-compiled statistics on Wikipedia. However, in cases where Wikipedia does not provide any candidates, we turn to BaseKB for candidate generation.

A sixth assumption, to be revised in future work, is that the information provided by the EME module is reliable. Therefore the CRL module does not attempt to adjust mention boundaries, entity types or mention types. It assumes the entity types and mention types are correct and uses them as constraints on the linking and coreference searches.

2.2.1 Generating candidate entities for NAM mentions

Even though the reference knowledge base is BaseKB, we opted to link first to Wikipedia, whenever possible, as Wikipedia provides access to two additional sources of information: the actual content of the Wikipedia pages, which we can use for contextual disambiguation, and its internal hyperlinks, which provide training data to estimate the probability $P(\text{page}|\text{surfaceForm})$ of a page given the surface form of an anchor (i.e. the text used in a page to refer to another Wikipedia

[3]We map all characters to ASCII using the Unicodedata library for Python (`https://docs.python.org/2/library/unicodedata.html`).

[4]We used the Punkt sentence splitter implemented in NLTK (`http://www.nltk.org/`).

[5]We first extract XML tags, then tokenize the text between tags using whitespace and punctuation as token boundaries.

[6]To differentiate surface forms from entities, we will use square brackets to mark entities in this section. This will be easier to read then referring to the entity by its BaseKB identifier.

page). Using a recent Wikipedia dump[7], we precomputed these prior probabilities for all anchor surface forms, which can then be used to generate likely candidate entities (i.e. pages) for a given mention.

The surface form we use to look up the precomputed $P(\text{page}|\text{surfaceForm})$ is very important, as small variations in the surface form can result in very different probabilities. To deal with these variations, when retrieving the prior probabilities computed using Wikipedia, we try different variations of the word (e.g. case variations). Also, to account for the fact that the words surrounding a mention can help in the search for candidates (e.g. *president* Obama), we build a compound around the mention (when possible) to have a more specific surface form with which to query Wikipedia. Rather than use a dependency parser to build this compound, we simply look for specific parts-of-speech (NN, NNS, JJ, NNP, NNPS) occurring one or more times before a given NAM mention.

As we gather candidate pages for different variations of the mention (e.g. uppercase, lowercase, first letter capitalized, within larger compound), we must combine the lists of candidates and their scores (prior probabilities). We do this by taking the maximum score of each page that appears in any of the lists.

If all queries fail using Wikipedia, we query BaseKB directly using a bag-of-words (BOW) search. For example, for "Department of Foreign Affairs", we would create the BOW {Department, of, Foreign, Affairs} and perform a search using a Lucene index of BaseKB to retrieve candidate nodes. These nodes do not have a prior probability like that of the Wikipedia pages. Therefore, to score the candidates, we measure the surface form similarity between a given node's labels and the mention, and we use the maximum label similarity to estimate that candidate node's prior probability. The string similarity measure is bigram overlap. We normalize the similarity measures according to the length of the mention. As long mentions tend to be less ambiguous, if we query the KB for a long mention and find a node with a very similar label (in terms of bigram overlap), we can be confident that we have found the correct node, but if the mention is quite short (5-6 characters), we should not be so confident.

[7]We used enwiki-20170720-pages-articles.xml, found at https://dumps.wikimedia.org/.

2.2.2 Coreference resolution of NAM mentions

Our next step is to build coreference chains (CCs) of NAM mentions, by assigning mentions that have similar surface forms to the same chain.

We look for four different types of surface form similarity:

1. Acronym expansion, e.g. "DFA" and "Department of Foreign Affairs". We simply look for mentions whose initials correspond to the acronym.

2. Abbreviation, e.g. "Gov." and "Government". We only consider single word abbreviations at the moment, but we should extend this to multi-words, e.g. "Dept. of Foreign Affairs".

3. Acronym variations, e.g. "D.F.A." and "DFA". We simply look for use of periods.

4. Matching head nouns, e.g. "Muslim Brotherhood" and "Brotherhood". We check if the head of one mention is the same as another mention.

As we create CCs containing multiple surface forms, e.g. "Muslim Brotherhood" and "Brotherhood", we must decide on the candidate pages that will be associated with each CC. Empirically, we see that multi-word queries will tend to be more specific than single-word queries. In our example, the candidate pages for "Muslim Brotherhood" should be trusted more than those for "Brotherhood". Likewise, the prior probabilities $P(\text{page}|\text{"Barack Obama"})$ should be trusted more than $P(\text{page}|\text{"Obama"})$. To combine these scores, we use the same maximum score strategy as described earlier to merge the candidates of different variations of a mention, but use an ad hoc weighting scheme to boost the scores of multi-word queries.

At this point, we've created a set of CCs (containing NAMs only), each associated with a set of candidate pages which were found by looking up various surface form variations of the mentions.

2.2.3 Type-based filtering

The candidate pages (or nodes) found in the previous steps can correspond to various types of entities: movies, electronic devices, songs, health issues, etc., most of which are not relevant to this task. As we consider the various candidates for a given CC, we wish to keep candidates which are of the same entity type as the one predicted by the EME module (PER, LOC, GPE, FAC or ORG). However, Wikipedia and BaseKB do not explicitly contain information about the five specific entity types which are relevant for the TEDL task.

Our strategy was to obtain statistics about the types which are used in BaseKB (based on the predicate *r_type*) to describe each of the 5 entity types targeted for this task. For each of these 5 entity types, we went through all the linked mentions of that type found in the 2016 EDL evaluation dataset (whether NOM or NAM), and gathered the set of *r_type* predicates (*r_types* for short) of their corresponding BaseKB nodes.

For example, the *r_types* for PER would include f_people.person, f_government.politician, f_book.author, and f_award.award_winner. And the *r_types* for LOC would include f_location.location, f_location.statistical_region, f_base.athletics.topic, f_business.employer, f_periodicals.newspaper_circulation_area, and f_meteorology.cyclone_affected_area. The *r_types* common to all nodes in the 2016 gold standard were excluded.

To filter candidates, we simply check whether any *r_type* of a candidate node is contained within the set of *r_types* associated with the predicted entity type of the mention. If so, we keep the candidate node, otherwise we discard it.

2.2.4 Local context-based candidate reranking

The previous steps of candidate generation and filtering provide an initial ranking based on the "commonness" of the candidate Wikipedia pages. At this point, we have a non-contextualized ranking of candidate pages for each CC. We now wish to rerank the candidates based on the similarity between the context of the mentions and the definitions of the candidate entities, as found in their Wikipedia page.

In accordance with the "one-sense-per-discourse" assumption, we take the full text to serve as context. A CC's context will be represented in two ways. First we generate a BOW b_1 from the words found within the set of CCs built so far, and a second BOW b_2 is generated from the words found within any of the document's mentions (including the NOM mentions, which have not been added to the CCs yet).

To represent the definitions, we also generate two different BOWs, one from the words in the first paragraph of the Wikipedia page and one from the words in the entire page.

As we have 2 possible representations for the context and 2 for the definition, we have 4 possible combinations, so we compute the 4 pairwise vector similarities (using cosine similarity) and take the maximum similarity score. We then adjust each candidate's score by weighting its original commonness score using the maximum context similarity: each candidate score S is replaced by $S' = S * (1 + \max(\text{sim}(\text{context}, \text{page})))$.

At this point, for each CC (containing NAMs only), we have a ranked list of candidate pages whose scores have been weighted based on context.

2.2.5 Adding NOMs to the coreference chains

So far, in accordance with the first assumption stated at the beginning of this section, we have focused on the NAM mentions, as these are explicit surface forms for specific entities. The current step aims at including the NOM mentions in the CCs.

Coreference between NAM and NOM mentions is resolved using three heuristics, based respectively on apposition, definitional markers and hypernyms.

1. First, we search for appositions (e.g. "Morsi, a man who ...") in the text. As we do not use a dependency parser, we use part-of-speech (POS) patterns to find appositions: we look for nouns appearing near the mention, allowing only specific POS to appear between the two, excluding POS such as verbs and conjunctions. For example, given "Mandela, a beacon for his people...", we would detect "beacon" as an apposition, and add it to the CC containing "Mandela", but "Mandela and a man who ..." would not establish coreference between "Mandela" and "man".

2. Second, we look for explicit definitional patterns, such as "Egypt is a country in which...", which should lead to "country" being added to the CC of "Egypt". In the current version of the system, we are barely exploring this idea as we've defined a single definitional pattern, "is a", but this idea could be extended. Furthermore, we require that coreferring mentions have matching entity types. This is useful for handling complex nouns.

For example, in "The president of Ukraine is a man who ..." we would want the definitional pattern to link "president" and "man" and not "Ukraine" and "man".

3. Third, we try to find a hypernymy link between a given NOM and any NAM in the text. For a NOM mention x (e.g. "country"), we look up the *r_types* of the surrounding NAM mentions to see if they contain x (e.g. a NAM such as "Egypt" having the *r_type* "country"). If so, we add x to the CC containing that NAM. This strategy is not as easy as it seems and it does produce its share of noise. The example above is actually oversimplified, as "country" would not actually occur, as is, as an *r_type*. Actual *r_types* have forms such as f_location.country. This *r_type* contains the word "country", so we can use this string inclusion to establish coreference. But the same strategy would link a NOM mention "organization" to a NAM mention having a *r_type* such as f_organization.organization_member which would be erroneous.

The first two rules attempt to connect two consecutive mentions, but the last one could consider any NAM in the text, in theory. We chose to use a concentric expansion search, whereby we first consider the closest NAM backward, then forward, then backward again, then forward again, and so on. We set a threshold on the number of steps to limit the search, as it becomes less and less likely that we will find a coreferring NAM as we move further away from the NOM mention. This limit was empirically set at 10 mentions before and after.

At this point, we have the same number of CCs as at the end of the previous step, but the CCs can now contain NOM mentions, not only NAMs. Yet there might remain some NOM mentions for which we found no coreferring NAMs. We look at those next.

2.2.6 Generating candidates for isolated NOMs

If any NOM query has not been added to a CC at this point, we try to figure out if it could independently be linked to its own entity. But generating candidates based on commonness as we did earlier for NAM queries would be useless. For example, if we search for "president", we will obtain a large number of candidates, all with small probabilities, except perhaps for the few presidents most frequently mentioned in Wikipedia (the US president for example). We need the queries to be contextualized before we search.

The strategy we use to attempt this contextualization is to construct noun phrases which contain the NOM mention and any surrounding NAM mention. Using one of a few prepositions (*in*, *of*) or the possessive marker (*'s*), we create alternate surface forms for the NOM mention which is thereby contextualized by the nearby NAM. For example, given the NOM "president" and the NAM "Egypt", we search for the query "president of Egypt".

As we do not know ahead of time which NAM we should use to construct these noun phrases, we try with all the NAM mentions within a certain distance of the NOM. Over-generating new forms will not harm the results, as it will simply not generate any new candidate pages. For example, if we have two nearby NAM mentions "Egypt" and "Cairo", we would search for "Egypt's president", "president in Egypt", "president of Egypt", "Cairo's president", "president in Cairo", "president of Cairo". Most of these constructed queries will not generate any candidate Wikipedia pages.

At this point, some NOM queries have generated their own candidates, and we proceed to the next step, in which we try to find coherence among the entities. No new candidates will be generated beyond this point.

2.2.7 Coherence-based reranking

We now try to establish a set of entities which have some degree of coherence between them. Both Wikipedia and BaseKB are used to measure coherence.

We first select a core set of KB nodes, based on the current scores of the candidate nodes (or pages). A node is promoted to the core nodes if it is the top-ranked candidate of one of the CCs and its score is greater than some threshold, which we tuned empirically.

From there, the overall strategy is to boost the score of candidate nodes which have some degree of coherence with the set of core nodes. This will influence the ranking of candidates for a given CC and possibly the top-ranked candidate.

Two measures of coherence are used, one based on the predicates in the KB, and the other on the text similarity of candidate Wikipedia pages:

1. KB coherence: If a candidate node has some predicates linking it to the core nodes, we consider that node to be more "connected" and therefore more coherent with the core nodes representing the entities in the text. A good strategy could be to boost a node's score based on its degree of connection. For now, a simpler winner-take-all approach is used and only the most connected candidate gets boosted.

2. Wikipedia similarity: Using Wikipedia, the degree of coherence is estimated based on text similarity. The more similar a given candidate's Wikipedia page is to the Wikipedia pages of the core nodes, the more its score gets boosted. We use cosine similarity, based on BOW representations of the full content of a given pair of Wikipedia pages. All candidate scores are weighted by their average similarity to the core nodes.

Once the candidate scores have been adjusted to account for coherence with the core set of nodes, we make one last attempt at merging CCs based on their candidates. If two CCs share a candidate, we merge them. For example, say a CC containing the query "president of Egypt" generated many candidate Wikipedia pages representing different presidents of Egypt at various points in history. If one of them also happens to be a candidate for another CC in the same text, that candidate would be favoured and the CCs would be merged.

2.2.8 Detecting and clustering NIL mentions

To detect NIL mentions, we use a threshold on the score of the top-ranked candidate of each CC, which we tuned on the 2016 EDL evaluation dataset.

To cluster NIL mentions, we first cluster NIL NAM mentions across documents based on string match and intra-document coreference, then we assign all the mentions in each NIL CC to the cluster of the NAM mentions they contain, if any. NIL NOM mentions which haven't been assigned to a cluster are then clustered across documents by string match. Discussion forum post authors are clustered separately, by string match within documents, but not across documents.

2.2.9 Parameter tuning

The CRL module includes multiple parameters that needed to be tuned. Given the time required to execute a single run (about 45 minutes to run the CRL module on the 2016 data), an extensive grid search was not feasible. Therefore ad hoc tuning was performed by varying one parameter at a time to assess its impact. This might not have lead to the optimal settings, and future work should focus on learning the various parameters. These parameters could include:

1. Entity type filtering: instead of enforcing a strict entity type matching requirement, we could penalize candidates that don't have a matching type using a parameterized weighting function.

2. Local context: the context size (sentence, paragraph, fixed number of words) to use when comparing a mention's context to a Wikipedia page influences the similarity score.

3. Maximum number of Wikipedia candidates: Since CPU power is not limitless, the number of candidates evaluated must be limited. Either a threshold on the prior probability $P(\text{page}|\text{surfaceForm})$ or a fixed number of candidates can be used.

4. NIL threshold: Under what threshold should we decide that the top-ranked candidate is not certain enough and that NIL is a better choice.

5. Core threshold: As we establish a set of core nodes as a basis for coherence measurements, what should be the minimum score of a node for it to be part of the core.

6. Contextualizing page probabilities: During local context-based reranking, we boost a candidate page's score using the function $S' = S * (1 + \max(\text{sim}(\text{context}, \text{page})))$, but the parameters of the reranking function could be learned. Also, the pairwise similarities of different representations of the context and the Wikipedia page are currently combined by taking the maximum similarity, but other functions could be used (or learned).

7. Polysemy-based weighting: When we merge the candidates and scores of coreferring mentions, multi-word queries are currently given twice the weight of single-word queries. For example, the candidates for "Barack Obama"

would receive twice the weight of the candidates for just “Obama”, as the first query is less polysemous. This weighting factor could be another learned parameter.

8. Search type: The type of search used to find coreferring NAMs for NOM mentions is currently an expanding concentric search. We also tried looking backward only, and empirically the bulls-eye seemed better, but this could be parameterized and learned. We limited the search to 10 mentions backward and forward, but this was set arbitrarily.

3 Dry runs

Before submitting, we evaluated our system on the 2016 EDL evaluation dataset, filtering the results to take into account English data only. Since the EME model is trained on the 2016 data (and, for 2 of the runs, the 2015 data), its predictions were produced using 3-fold cross-validation on the 2016 data. This was meant to produce a less biased estimate of the system's accuracy, but we assumed that the scores obtained would be somewhat optimistic nonetheless.

Run	NERC	NERLC	CEAFmC
1	0.820	0.655	0.696
2	0.821	0.651	0.695
3	0.807	0.648	0.689
4	0.801	0.645	0.686

Table 1: F-scores on the 2016 evaluation dataset (English only)

A summary of the results is presented in Table 1. The runs are ordered as follows:

1. N-gram classifier with transfer learning
2. CRF with transfer learning
3. N-gram classifier without transfer learning
4. CRF without transfer learning

Our best scores were obtained using the transfer learning approach, with both classifiers producing similar scores.

4 Results

For our first participation in the TEDL task, we submitted results for English only. A summary of the results of the 4 runs we submitted is shown in Table 2. These results were filtered to take into account English documents only. The table shows not only the results of the 4 runs we submitted, but also the result of an ensemble method we tested after the evaluation period: we took all the mentions that were predicted in at least 2 of the 4 runs (taking a vote in cases where different entity or mention types were predicted by different models), then applied our linking module to this set of mentions. This increased our entity mention extraction score (NERC f-score) by almost two points, but had little effect on the scores that take into account the linking or clustering accuracy (NERLC and CEAFmC).

Run	NERC	NERLC	CEAFmC
1	0.764	0.610	0.668
2	0.742	0.580	0.642
3	0.747	0.587	0.646
4	0.740	0.582	0.638
ensemble	0.782	0.610	0.671

Table 2: F-scores on the 2017 evaluation dataset (English only)

If we focus on the results of our 4 submitted runs, they show that our dry run results were indeed somewhat optimistic. The overall accuracy (i.e. NERC f-score) of our entity mention extraction module was several points lower on the 2017 data, mainly due to lower recall. This in turn lowered the evaluation measures that take into account linking (NERLC) and clustering (CEAFmC).

As expected based on our dry runs, the best results were obtained using the n-gram classifier. Transfer learning seems to have increased the accuracy of this model somewhat (compare runs 1 and 3), but does not seem to have helped the model which exploits a CRF (runs 2 and 4).

The difference between the 4 runs is limited to about 3 points, which suggests that the type of classifier (n-gram classifier vs. CRF) and the use of transfer learning did not have a huge impact on the results.

4.1 Breakdown

A detailed breakdown of the results of our best run (run 1) is shown in Table 3.

These results illustrate a few trends which were common to all of our submitted runs:

- The detection (NERC) scores are somewhat higher on discussion forums (DF) than

Filter	NERC	NERLC	CEAFmC
DF	0.771	0.682	0.706
NW	0.758	0.548	0.640
NAM	0.819	0.702	0.778
NOM	0.597	0.332	0.429
FAC	0.418	0.291	0.331
GPE	0.832	0.748	0.776
LOC	0.483	0.294	0.357
ORG	0.673	0.459	0.563
PER	0.836	0.672	0.725

Table 3: F-scores of our best run, filtered by text type, mention type, and entity type (English only)

newswire (NW) texts, but the scores that take into account linking (NERLC) and clustering (CEAFmC) are quite a bit higher, which suggests that these sub-tasks are harder on newswire texts. This might be due to the fact that a large portion of the gold mentions in the DF texts are post authors, which are easy to detect. Our simple regex matched 960 mentions in the 2017 data. Assuming this is close to the exact number of post author mentions, this represents about 30% of all the gold mentions in the DF texts in English. There is also a greater number of unique entities (KB nodes) mentioned in the NW texts (550 vs. 380), so perhaps there are more obscure or emerging entities in NW texts.

- Results are much better on named (NAM) mentions that on nominal (NOM) mentions, and the difference is particularly important as regards linking and clustering accuracy (NERLC and CEAFmC). That being said, even though our results seem to be good on NAMs and bad on NOMs, if we compare the results of our best run on NOMs (run 3) to those of the other systems that were evaluated on the English EDL data, our results are very good (relative to other participants), being ranked 2nd in terms of NERLC and CEAFmC. Therefore, our results are relatively good on NOMs, which are harder to detect, cluster, and link, but there is room for improvement on NAMs.

- The entity types that our system handles best are PER and GPE, followed by ORG. Accuracy on LOC and FAC entities is much lower.

Although run 1 produced the best results globally, other runs sometimes produced better results on specific subsets of test cases. Run 1 was best on both NW and DF texts, and on NAM mentions, however:

- Run 3 was best on NOM mentions.
- Run 3 produced a slightly higher NERC f-score on LOC mentions.
- Run 4 was best on FAC mentions.
- Run 2 produced higher NERC and CEAFmC f-scores on both ORG and PER mentions.

This suggests that we might obtain better results by using different models to handle different types of mentions or entities, perhaps using multi-task learning.

The difference in accuracy on different types of mentions and entities may be explained, at least in part, by the class frequency distribution of the training data. For instance, in the 2016 dataset, over 75% of the gold mentions in English documents are NAM mentions, which might explain to some degree why accuracy is lower on NOM mentions. Moreover, the entity types that our system handles best (PER and GPE) are the 2 most frequent types in the 2016 data, representing 36% and 29% of the mentions respectively, whereas LOC and FAC mentions represent only 6% and 4% of the gold mentions respectively.

4.2 Nested mentions

As we explained in Section 2.1.2, the reason we experimented with the n-gram classifier was to allow for nested mentions. The fact that the results obtained using this classifier were not much better than those produced by the CRF tagger is explained by the low number of nested mentions in the gold standard. By our count, the gold standard contains 182 mentions which are nested in a larger mention (in English documents), such as [[Yorkshire] Ripper] or [[Asia]-[Pacific]], which contains 2 nested mentions. This represents only 2.6% of the gold mentions in English, which is close to the gain in accuracy we achieved by using the n-gram classifier rather than a CRF.

The n-gram classifier is indeed capable of predicting nested mentions, with run 1 producing 122 nested mentions, and run 3 producing 197. However, given that there are few nested mentions in the gold data, this did not have a huge impact on our scores. Also, recall that we set the maximum n-gram size to 5, so longer mentions were not detected, but these appear to be very rare.

Furthermore, the n-gram classifier sometimes predicts mentions that are not completely contained in, but partially overlap another mention, such as “United Nations” overlapping with “Nations Higher Council”. These cases are extremely rare, with only 2 appearing in the output of run 1 and 8 for run 3. We found no such cases in the gold standard.

4.3 Error analysis

By analyzing the errors made by our system (in run 1), we were able to identify a few common types of errors regarding NERC and entity linking, which we will outline in the following sections.

4.3.1 Named entity recognition and classification

If we consider only the spans of the predicted and gold mentions, we see that we have 1739 false negatives and 776 false positives. If we look at the predicted mention type and entity type of the mentions that the system detected correctly (in terms of span), we see that we have 260 misclassifications.

False negatives The 2 strings that we fail to detect most often are “here” and “there”, which were often annotated (37 and 30 times respectively) in the gold standard as nominal mentions. We should note that these are adverbs and not nouns, and no occurrences of these words were annotated as mentions in the 2016 data, which we used for training. The next 3 are “one”, “world”, and “school”, nominal mentions which our system detects in some contexts, but not all. It seems our model was not able to learn in a satisfactory way in which contexts these words refer to a specific entity. This may be due to the fact that the model only takes the sentence as context.

False positives The 6 strings that we detect erroneously most often are “National” (14 errors), “company” (12), “court” (12), “committee” (11), “state” (11), and “country” (11). The first is an interesting case: in our training data, it was often annotated as a nominal mention (referring to a specific country) nested in larger mentions, such as “National Assembly” or “National Security Agency”, but in the 2017 evaluation data, we find no annotated occurrences of “National”, although we do find mentions such as “National Assembly” and “National Rifle Association”. This suggests either inconsistent annotations or an explicit change in annotation policy. As for the other five words, these are sometimes annotated as nominal mentions in the gold standard, but our system sometimes detects them erroneously, i.e. in contexts where they do not refer to specific entities. If we look specifically at the word “company”, our system detected it 84 times (which is approximately every time that word occurs), and was wrong in 12 cases. As it happens, 11 of these errors were in discussion forums, in contexts such as “I could see him taking on a new company”. It seems plausible that such *hypothetical* entities are more frequently mentioned in discussion forums than in news articles, but we have not investigated this extensively.

Misclassifications Finally, if we look at the misclassifications, we also find a case that suggests inconsistencies between the 2016 and 2017 annotated data. The most frequently misclassified mention is “West”, with 9 errors. This mention was systematically annotated as a LOC in the 2016 data, which we used for training. Therefore, our model predicts this class, but in the 2017 gold data, this mention is systematically annotated as a GPE. The second most frequently misclassified mention is “Asiana”, which our model classifies as a GPE, although it is an ORG (specifically, an airline). We suspect our model thinks it is a GPE because of the word’s morphology: our model never saw this word in the training data, but often saw similar words such as “Asia” and “Asian”, and possibly used this information (as well as case and context features) to detect the mention “Asiana”, even though it had never seen it before. However, it guessed the wrong entity type.

4.3.2 Entity linking

Analyzing the errors made by the linking module leads us to conclude that several of the heuristics implemented in this module produced errors at one point or another. Below are some examples of errors to illustrate what the algorithm misses or confuses.

Coreference resolution errors A number of errors were due to NOM mentions for which no coreferring NAM mentions were found, which were thereby classified as NIL, including mentions such as “mall”, “country”, “government”, “leader” or “group”. Improving coreference resolution would help link these mentions which were incorrectly labeled NIL. There were also a few errors due to incorrect (rather than missing) coreference links. Examples include “company” (6 occurrences) being linked to [Apple Inc.] instead of [Foxconn], “region” (4) being linked to [Pacific Ocean] instead of [Asia-Pacific], and “government” (3) being linked to [Communist Party of China] instead of [Government of China].

Surface form variations Although we used a few heuristics to capture surface form variations of names, one that we seem to have missed is the use of first names to refer to people previously mentioned in text. For example, we missed “Steve” (17 occurrences) referring to [Steve Jobs] and “Tom” (7) referring to [Tom Merrit].

Typing heuristic To filter the candidate KB nodes, we used a heuristic which consisted in making sure that at least one *r_type* of a candidate node was included in the *r_types* associated with the entity type predicted by the EME module (see Section 2.2.3). This filter let through some noise, as the linking module sometimes selected nodes that do not actually belong to any of the 5 target entity types. For example, the mention “Sheng” (4 occurrences) was linked to [Sheng (instrument)] instead of [Sheng Guangzu], and the mention “Marty” (2) was linked to [Marty (film)] instead of [Marty Walsh].

Geographical ambiguity Some errors were due to confusion between cities and states. For example, the mention “New York”, detected 18 times as a geopolitical entity (GPE), was linked to [New York] rather than [New York City], and the mention “Washington” (5 occurrences) was linked to [Washington state] instead of [Washington D.C.].

Close match In many cases, the predicted node is actually quite close to the correct node even though it is considered incorrect. For example, the mention “Samsung” (9 occurrences) was linked to [Samsung Group] rather than [Samsung Electronics], the mention “BlackBerry” (3) was linked to [BlackBerry] (the brand) rather than [BlackBerry Ltd] (the company), the mention “White House” (9) was linked to [White House] instead of the [Executive Office of the President of the United States], and the mention “administration” (2) was linked to [Presidency of Barack Obama] instead of [Executive Office of the President of the United States]. Errors such as these may be explained, at least in part, by the similarity of the candidates’ Wikipedia pages. In this close match category, one mention was a very frequent source of errors: the mention “Xinhua”, detected 55 times as an organization (ORG), was linked to [Xinhua Holding] instead of [Xinhua News Agency].

5 Discussion

It is interesting to note that the best linking and clustering scores achieved by the various participants on the English EDL data this year have not increased much compared to last year, whereas the entity mention extraction scores have increased by several points. This is likely due to the fact that training data were available for all classes (entity types and mention types), whereas last year, training data for the NOM mentions were only available for PER entities. The fact that linking and clustering scores have not improved may indicate that the entity mentions which are more accurately detected now are hard to link and cluster. As it happens, our system achieves strong results, relative to other teams, on the detection, clustering, and linking of NOM mentions.

It is important to remember that our approach to entity mention extraction exploits machine learning almost exclusively, therefore any comparison to systems that exploit gazetteers or other language or knowledge resources should take this into consideration. Furthermore, we only exploited training data from the 2015 and 2016 editions of the TEDL task, and only submitted runs produced by single models, no ensembles.

Improvements to the entity mention extraction system could include the incorporation of feedback from the linking module, which would allow us to exploit information from the KB. We could also consider allowing the n-gram classifier to handle arbitrarily long mentions, rather than setting a fixed maximum size for the n-grams that are classified. It would also be interesting to take into account a wider context, this being limited to the sentence for the moment.

As for the coreference resolution and linking module, we believe improvements could be achieved by integrating all the different strategies and heuristics within a reinforcement learning framework. We emphasized earlier that quite a few parameters need to be tuned within the CRL module, and in the future we should try learning these parameters.

6 Concluding remarks

To tackle the task of entity detection and linking, we explored both deep learning methods and more ad hoc methods based on corpus statistics and linguistic heuristics. The n-gram classifier we developed to detect entity mentions was effective at handling nested mentions, and transfer learning allowed us to improve our system's performance by exploiting more training data. As for the task of linking entity mentions to the knowledge base, focusing on coreference resolution proved to be an effective strategy. By analyzing the mistakes made by our system, we were able to identify a few common errors, including some due to inconsistencies between the training and testing data. Overall, our system performed well, especially on nominal entity mentions, which are harder to detect, cluster, and link than named mentions.

Acknowledgments

This research was carried out as part of the PACTE project (Ménard and Barrière, 2017), and was supported by CANARIE and the *ministère de l'Économie, de la Science et de l'Innovation* (MESI) of the Government of Québec.

References

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR* abs/1412.6980. http://arxiv.org/abs/1412.6980.

Pierre André Ménard and Caroline Barrière. 2017. PACTE: A collaborative platform for textual annotation. In *Proceedings of the 13th Joint ISO-ACL Workshop on Interoperable Semantic Annotation (ISA-13)*. pages 95–99.

Avirup Sil, Georgiana Dinu, and Radu Florian. 2016. The IBM systems for trilingual entity discovery and linking at TAC 2016. In *Proceedings of the Ninth Text Analysis Conference (TAC 2016)*.

Mingbin Xu, Feng Wei, Sedtawut Watcharawittayakul, Yuchen Kang, and Hui Jiang. 2016. The YorkNRM systems for trilingual EDL tasks at TAC KBP 2016. In *Proceedings of the Ninth Text Analysis Conference (TAC 2016)*.