Overview of the Ugglan Entity Discovery and Linking System

Marcus Klang **Firas Dib** **Pierre Nugues**
Lund University
Department of Computer Science
S-221 00 Lund, Sweden
marcus.klang@cs.lth.se pierre.nugues@cs.lth.se
firas.dib@gmail.com

Abstract

Ugglan is a system designed to discover named entities and link them to unique identifiers in a knowledge base. It is based on a combination of a name and nominal dictionary derived from Wikipedia and Wikidata, a named entity recognition module (NER) using fixed ordinally-forgetting encoding (FOFE) trained on the TAC EDL data from 2014-2016, a candidate generation module from the Wikipedia link graph across multiple editions, a PageRank link and cooccurrence graph disambiguator, and finally a reranker trained on the TAC EDL 2015-2016 data.

In our first participation in the TAC trilingual entity discovery and linking task, we obtained a *strong typed mention match* of 0.701 (Ugglan2), a *strong typed all match* of 0.592 (Ugglan4), and *typed mention ceaf* of 0.595 (Ugglan1).

1 Introduction

The goal of the trilingual entity discovery and linking task (EDL) in TAC 2017 (Ji et al., 2017) is to recognize mentions of entities in Chinese, English, and Spanish text and link them to unique identifiers in the Freebase knowledge base. In the TAC datasets, the mentions have a type, either persons (PER), geopolitical entities (GPE), organizations (ORG), locations (LOC), or facilities (FAC), and their syntactic form can consist of proper or common nouns, called respectively named and nominal mentions. Some entities in the annotated corpus are not in Freebase. They are then linked to a NIL tag and clustered across the three languages; each specific entity being assigned a unique identifier.

In this paper, we describe Ugglan, a generic multilingual EDL platform that required minimal adaptation to the TAC 2017 tasks. We detail the system architecture and its components as well as the experimental results we obtained with it.

2 System Overview

Ugglan has a pipeline architecture that consists of three main parts:

- A mention discovery that uses a finite-state automaton derived from Wikipedia and/or a feed-forward neural network trained on the TAC 2014-2016 data (Ji et al., 2014, 2015; Ji and Nothman, 2016);
- An entity linker that uses mention-entity pairs extracted from Wikipedia and ranks them using PageRank;
- A reranker trained on the TAC 2014-2016 data.

The Ugglan architecture is modular and parameterizable, and its parts can use independent algorithms. To build it, we used a set of resources consisting of Wikipedia, Wikidata, and DBpedia.

2.1 Mention Discovery

The first part of the processing pipeline is the discovery of mentions of entities in text. It starts with a custom multilingual rule-based tokenization of the text and a sentence segmentation. We then normalize the letter case based on statistics from all the Wikipedia pages.

We discover the mentions using a combination of a finite-state transducer (FST) built from mentions extracted from Wikipedia and an optional named entity recognizer (NER) based on neural networks. As an alternative, Ugglan can also use the Stanford NER (Finkel et al., 2005).

The mention discovery results in an overgeneration of mention candidates. For instance, the phrase

United States of America

results into as many as eight candidates; see Fig. 1. We prune them using parameterized rules.

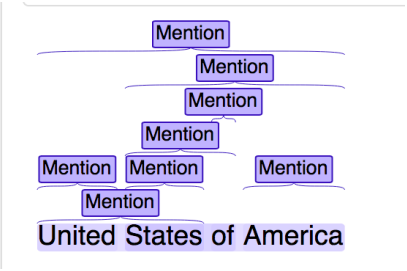

Figure 1: Mention candidates produced by the finite-state transducer for the phrase *United States of America*

Finally, our system does not output overlapping mentions. We resolve this overlap using a statistical estimation of the mention "linkability": The **link density**; see Sect. 4.2. Figure 2 shows the overall mention detection steps.

2.2 Entity Linking

Once we have carried out the detection, we associate each mention with entity candidates by querying a mention-entity graph.

Some of the entities are referred by mention variants along a text, for instance starting with "Barack Obama" and then "Obama" or "Barack". We augment the entity recall by sorting the mentions in a document with a partial ordering using the `is_prefix` or `is_suffix` relations so that we have:

$$\text{Barack} \prec \text{Barack Obama} \quad \text{and} \quad \text{Obama} \prec \text{Barack Obama}.$$

Using this ordering, we expand the candidates of the preceding strings by adding the candidates of the including strings further up in the order. For instance, we add all the candidate entities of "Barack Obama" to the candidates of "Obama".

We extracted the graph of mentions to candidate entities from Wikipedia as well as the graph of entity–entity cooccurrences. We built this graph from outlinks gathered from the combination of seven Wikipedia editions.

Finally, we disambiguate the text entities using a local graph of candidates, on which we apply the PageRank algorithm. For each mention, PageRank assigns a weight to the candidates that enables us to rank the entities.

2.3 Reranking and Classification

After the entity linking step, each mention has a ranked list of entity candidates. We rerank these lists using a multilayer neural network trained on the TAC2015-16 data. This also results in some entities being assigned the NIL identifier.

We assign a type to the entities using a predefined dictionary mapping derived from DBpedia; this type is possibly merged with that obtained from the NER, if no available mapping exists.

At this point, we have discovered and resolved the named expressions. We apply a discovery to the nominal expressions (NOM) using a dictionary collected from Wikidata and a coreference resolution based on exact string matches.

Finally, we discard the classes not relevant to the TAC task.

3 Building Ugglan's Knowledge Base

Ugglan relies on a graph of mention-entity and entity-entity for both the discovery and the linking stages. We constructed this knowledge base from a set of resources:

- Seven Wikipedia editions: en, es, zh, de, fr, ru, and sv;
- Wikidata, which binds these editions together;
- DBPedia, which we used for the class mapping.

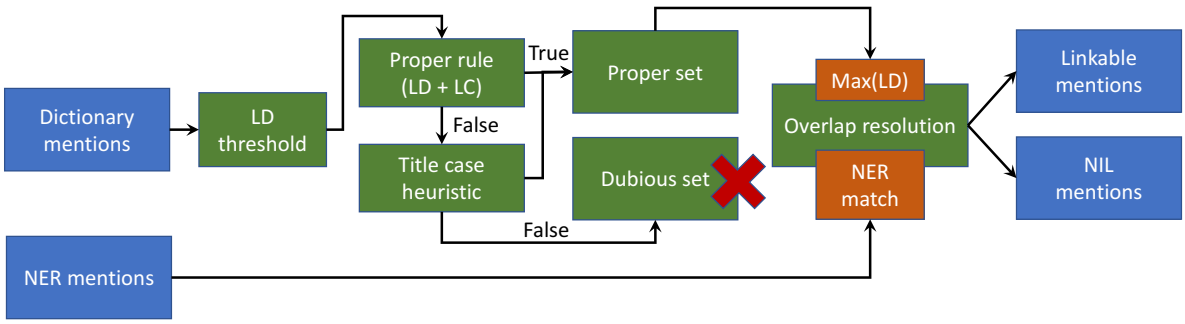

Figure 2: Overview of the mention pipeline

3.1 The Wikipedia Corpus

The Wikipedia corpus is our first resource from which we extract the text, the mentions, and link graph. We can access its content in multiple ways and with varying degrees of accuracy.

A simple approach is to download a XML dump, apply some filtering to the raw wiki markup, and then use this output as text. However, Wikipedia features many templates that are language dependent. Using a dump approach would leave the template expansion unsolved, as well as the local context of the links and information about the logical structure.

We opted therefore to use a rendered HTML version which expands the templates and has its Scribuntu scripts executed. This produces a HTML dump of Wikipedia, which is as true to the user online version as possible without actually replicating the full Wikipedia infrastructure. Concretely, we used a combination of Xowa (offline Wikipedia reader) and the public Wikipedia REST API to get the HTML dumps.

3.1.1 HTML Annotation Processing

The raw HTML must be filtered and simplified to be easy to process. We converted the hierarchical structure using a DOM tree produced by Jsoup and applied heuristics to produce a flattened version with multiple layers of annotations using the Docforia structure (Klang and Nugues, 2017). These layers include information on anchors, headers, paragraphs, sections, lists, tables, etc.

The Chinese version was processed in a special way as it can be translated in multiple variants: simplified, traditional, and localizations. To get coherent statistics, we reimplemented the translation mechanism used by the online version to produce a materialized `zh-cn` Wikipedia edition.

We finally resolved all the pages and anchors in the flattened version to Wikidata, which produces multilingual connections for most entities.

3.2 Multilingual Resources: Wikidata and DBpedia

Wikidata is Ugglan's repository of multilingual entities as it contains clean mappings between the multiple language editions, as well as detailed structured data. TAC uses Freebase as knowledge base and we created mappings to translate Freebase entities to Wikidata. Wikidata entities are represented by unique identifiers called Q-numbers. We mapped the Wikidata entities missing from Freebase to NIL-xx identifiers, where xx is a number unique for the entity.

We chose DBpedia to map entities to TAC classes as it produced subjectively better mappings than using Wikidata. Wikidata would have required additional rules to carry out the conversion.

3.3 Mention–Entity Graph

We associated each mention in Wikipedia with a set of potential entities in text. We used a dictionary, where the entry (or key) is the entity mention from Wikipedia and the value is the Q-number. In Wikipedia, we extracted the mentions from five sources across the languages we consider:

The title of the entity's Wikipedia page with and without parentheses. For instance, we

have $M_{Q90} = \{\text{Paris}\}$ and $M_{Q167646} = \{\text{Paris, Paris\_(mythology)}\}$.

The Wikipedia redirects: i.e. page names that automatically redirect to another page like the *EU* page that moves the reader to *European Union*. This enables us to collect alternative names or spelling variants so that we expand the mention list for the European Union entity, M_{Q458}, from $\{\text{European Union}\}$ to $\{\text{European Union}, \text{EU}\}$.

The disambiguation pages, where a page title is associated to two or more entities. The English Wikipedia has a disambiguation page associated with *Paris* that links to about 100 entities, ranging from the capital of France to towns in Canada and Denmark as well as song and film titles;

The wikilinks. A link in Wikipedia text is made of a word or a phrase, called the label, that shows in the text, and the page (entity) it will link to, where the wiki markup syntax uses double brackets: `[[link|label]]`. When the link and the label are different, the label is often a paraphrase of the term. We therefore consider all these labels as candidate mentions of the entity. Examples in Swedish of such labels for the former Swedish Prime Minister Göran Persson, Q53747, include the name itself, *Göran Persson*, 468 times, *Persson*, 14 times, and *Han Som Bestämmer*, 'He Who Decides', one occurrence.

The first bold-faced string. We finally used the first bold-faced string in the first paragraph of an article as it often corresponds to a synonym of the title (or the title itself).

3.4 Statistics

During the mention gathering, we also derive statistics for a given language. Before we compute these statistics, we apply a procedure that we called *autolinking*. In an article, the Wikipedia guidelines advise to link only one instance of an entity mention[1]: Normally the first one in the text. With the autolinking procedure, we link all the remaining mentions provided that we have sequences of exactly matching tokens. The statistics we collect are:

- The frequency of the mention string over the whole Wikipedia collection (restricted to one language);
- The frequency of the pair (mention, entity) that we derive from the links without autolinking (only manually linked mentions);
- The count of (entity1, entity2) pairs in a window corresponding to a paragraph and limited to 20 linked mentions. This is carried out after autolinking;
- Capitalization statistics for all the tokens: We extract token counts for all tokens with a frequency greater than 5 and we break them down by case properties: uppercased, lowercased, titlecased, and camelcase;

4 Mention Recognition

Ugglan uses a multilingual rule-based tokenizer and segmenter that we implemented using JFlex. For logographic languages such as Chinese, the tokens are equivalent to characters. The parser was customized to accept a mixture of both logographic and alphabetic text.

The tokenizer was used in conjunction with the Lucene analyzers. Lucene provides an infrastructure consisting of common filters and normalizers for many languages, from which we use case folding, accent stripping, Unicode form normalization, and stemming. These pipelines are configurable and easy to adapt for new languages.

The mention discovery is carried out by two modules: A dictionary-based finite-state transducer and a named entity recognizer (NER) using neural networks that we trained on the TAC data; see Sect. 5. These two modules can work in tandem or independently.

[1] https://en.wikipedia.org/wiki/Wikipedia:Manual_of_Style/Linking#Overlinking_and_underlinking

The mention recognition pipeline has three primary steps: discovery, filtering, and overlap resolution. In addition, the discovery pipeline can be configured to use one of three modes: NER-only, dictionary-only, and a hybrid mode. The primary difference between these modes is how the filtering and overlap resolution operates. Figure 2 shows an overview of these steps.

4.1 Discovery

Before querying the FST dictionary, we normalize the tokens in uppercase or lowercase characters for English and Spanish using statistics derived from Wikipedia. For instance, we convert *BEIJING* into the title case variant *Beijing*. However, due to ambiguity, we did not apply case normalization to title-cased words.

4.2 Filtering

The mentions of named entities are likely to be linked in Wikipedia. Examining the articles, we observed that, given word sequence, the relative frequency of linkage often reveals its ambiguity level. For instance, while the word *It* can refer to a novel by Stephen King, it is rarely an entity and, at the same time, rarely linked.

The *link density* (LD) is a measure derived from the analysis of text linkage in Wikipedia. It loosely corresponds, as the original text is not fully linked, to the probability of a sequence of tokens being linked in the source edition.

We estimated the linkage probability by applying the FST dictionary to Wikipedia in an offline step. We counted the exact matches with the known ground truth: The anchors created by the Wikipedia editors. In addition, before counting, we added the *autolinked* anchors that matched existing ones perfectly:

$$\text{Link density} = \frac{\#\text{Anchor}}{\#\text{Text} + \#\text{Anchor}} \qquad (1)$$

In addition to link density, we used the gold standard mention counts, the *link count* (LC), as a measure of significance.

Before the overlap resolution, all the mentions from the dictionary are classified into either a proper set or a dubious set. The proper set consists of all the mentions which exceed LD and LC thresholds; The proper set also includes the mentions which do not exceed these thresholds if at least 75% of the tokens in the mention are title cased.

4.3 Overlap Resolution

The mentions placed in the *proper set* and the mentions found by the NER will be merged and resolved in the overlap resolution step. The NER module itself only outputs nonoverlapping mentions. This stage works differently depending on which mode is used:

- *NER-only* accepts only NER mentions and produces linkable mentions, if an exact dictionary match is found.
- *Dictionary-only* ignores the NER mentions and solves overlapping mentions by picking the mention which has the largest LD value until no overlap exists.
- *Hybrid* merges NER mentions with dictionary matches by trusting the NER output where applicable i.e. when multiple candidates exist, it chooses the NER output, otherwise the dictionary output.

5 Named Entity Recognizer

We developed a named entity recognizer (NER) based on a feed-forward neural network architecture and a *fixed ordinally-forgetting encoding* (FOFE) (Xu et al., 2017; Zhang et al., 2015). This NER is part of the mention recognition module; see Figure 2 for the dataflow.

The NER operates over sentences of tokens and outputs the highest probability class using a moving focus window with varying width. The focus window represents potential named entity candidates and ranges from one to seven words. A more detailed explanation of this, and why there is an upper limit, is explained in Sect. 5.3

The NER can recognize both named and nominal expressions and predict their class. The named or nominal types are just extensions to the classes. If there were N classes originally, there would

be $2N$ outputs if all nominal classes were included. In the TAC2017 Ugglan system, the possible classes are:

- {*PER, GPE, ORG, LOC, FAC*}*-NAM* and
- {*PER, GPE, ORG, LOC, FAC*}*-NOM*.

The NER is identical in its construction for English and Spanish, without any language specific feature engineering. However, we modified this module for Chinese since the word segmentation was not found reliable. In Chinese, we used the individual characters (logograms) as word features and none of the corresponding Latin character features. In any case, the Chinese character features would be a subset of the word features.

5.1 The Fixed Ordinally-Forgetting Encoding

We applied a *fixed ordinally-forgetting encoding* (FOFE) (Xu et al., 2017; Zhang et al., 2015) as a method of encoding variable-length contexts to fixed-length features. This encoding method can be used to model language in a suitable manner for feed-forward neural networks without compromising on context length.

The FOFE model can be seen as a weighted bag-of-words (BoW). Following the notation of Xu et al. (2017), given a vocabulary V, where each word is encoded with a one-hot encoded vector and $S = w_1, w_2, w_3, ..., w_n$, an arbitrary sequence of words, where e_n is the one-hot encoded vector of the nth word in S, the encoding of each partial sequence z_n is defined as:

$$z_n = \begin{cases} 0, & \text{if n = 0} \\ \alpha \cdot z_{n-1} + e_n, & \text{otherwise,} \end{cases} \quad (2)$$

where the α constant is a weight/forgetting factor which is picked such as $0 \leq \alpha < 1$. The result of the encoding is a vector of dimension $|V|$, whatever the size of the segment.

Zhang et al. (2015) showed that we can always recover the word sequences T from their FOFE representations if $0 < \alpha \leq 0.5$ and that FOFE is almost unique for $0.5 < \alpha < 1$. Zhang et al. (2015) make the assumption that the representation is (almost) always unique in real texts.

5.2 Features

The neural network uses both word and character-level features. The word features extend over parts of the sentence while character features are only applied to the focus words: The candidates for a potential entity.

5.2.1 Word-level Features

The word-level features use bags of words to represent the focus words and FOFE to model the focus words as well as their left and right contexts. As context, we used all the surrounding words up to a maximum distance, defined by the floating point precision limits using the FOFE α value as a guide.

Each word feature is used twice, both in raw text and normalized lower-case text. The FOFE features are used twice, both with and without the focus words. For the FOFE-encoded features, we used $\alpha = 0.5$.

The beginning and end of sentence are explicitly modeled with `BOS` and `EOS` tokens, which have been added to the vocabulary list.

The complete list of features is then the following:

- Bag of words of the focus words;
- FOFE of the sentence:
  - starting from the left, excluding the focus words.
  - starting from the left, including the focus words.
  - starting from the right, excluding the focus words.
  - starting from the right, including the focus words.

This means that, in total, the system input consists of 10 different feature vectors, where five are generated from the raw text, and five generated from the lowercase text.

5.2.2 Character-level Features

The character-level features only model the focus words from left to right and right to left. We used two different types of character features: One that models each character and one that only models

the first character of each word. We applied the FOFE encoding again as it enabled us to weight the characters and model their order. For these features, we used $\alpha = 0.8$.

In order to ensure the characters fall into an appropriate range, we encoded them with a simple modulo hash. Each characters ASCII value is normalized to be within the range 0 and 128. This limitation is reasonable since most characters of English and Spanish are in the ASCII table. The Spanish characters in the range 128:256 are confused with unaccented ASCII characters, for instance $\tilde{n}$ with q.

5.2.3 Projection Layers

Characters. The character features are generated first as sparse one-hot encoded vectors of dimension 128 and then projected to a dense representation of lower dimension: 64. To project the character features, we used a randomly initialized weight matrix, which is trained as part of the network. This procedure produced better results than the direct input of one-hot vectors.

Words. We projected the word-level features to a 256-dimension dense representation. We initialized the projection layer with two different word2vec (Mikolov et al., 2013) models that we pretrained on the en, es, and zh wikipedias. One model was trained on normalized text, while the other was trained on the raw untouched text. These are incorporated into the rest of the network and consequently trained as a part of it.

When creating the word projection layers from the word2vec models, we used the weights of the top 100,000 words. We disregarded all the other words and instead mapped them to a *unknown word vector*. More specifically, for the case-sensitive projection layer, we return a `UNK` vector when we encounter a word with no embedding; if this word is equal to its normalized lower cased variant, we return a special `unk` vector instead.

5.3 Named Entity Candidates

The potential named entity candidates are produced by looping over each word in the sentence with a moving window that expands up to seven words. This exhaustively generates all the possible candidates in the sentence, which in turn produces a lot of noise. In the training process, we sample this noise to build a set of negative examples and instruct the network how to discriminate mention boundaries and invalid mentions.

The upper bound of seven was found by going through the annotations of the TAC 2016 data and seeing if there was any clear cut-off where results would start to diminish. After seven words, we found there was little benefit to go any further. This upper limit value is significant because each candidate which is not a positive sample is considered negative and in turn used in the training process.

A large focus window results in many negative samples, which are not representative of the real world. As the negative candidates are randomly sampled, we would (to some degree) get a skewed distribution of the negative samples. If, for example, the upper bound was set to 12 words, there would be many negative 12-token long samples in comparison to how many positive examples there are. We attempted to weigh the different selections with respect to the positive mention count. We set it aside for the TAC 2017 evaluation due to a lack of time.

In total, we considered three different cases to create the training data:

1. The mention candidate matches exactly an annotated sequence;

2. The candidate is completely disjoint, i.e., contains no annotated words;

3. The candidate partially or completely overlaps with an annotated sequence or is a subset of the sequence,

where an annotated sequence corresponds to all the words annotated with a given class in the data, such as `University of Lund`.

The first case corresponds to the positive examples that we label with the TAC classes, while the two last cases are the negative examples that we label as `NONE`. We will keep this stratification in the training step.

5.4 Training

In the data set we collected, the negative samples outnumber the positive ones by an order of magnitude. We used a manually-specified distribution of the samples to mitigate this unbalance and train the network. At the beginning of each epoch, the data is shuffled and the negative samples are reselected according to the distribution. This means that we continuously introduce new negative examples and previously unseen data to the network, which helps with regularization.

Table 1 shows the distribution we used for the TAC 2016 EDL task.

Candidate type	Ratio of sample size
Negative: Overlapping	60%
Negative: Disjoint	30%
Positive	10%

Table 1: The distribution between positive and negative mention candidates.

We trained the NER system with data from TAC 2014-15 and evaluated it on the 2016 data; see Table 2.

5.5 Neural Network Architecture

The network architecture can be conceptually divided into two parts: A first part projects the input features into a dense space and a second one classifies the input and outputs a class (see Figure 3). The classification part of the network consists of three hidden layers, which have batch-normalization layers sliced in-between them, and a final layer that outputs multiclass predictions.

During the development, we tested and evaluated several hyperparameters using a grid search method. Table 3 shows the final hyperparameters used in the TAC2017 EDL task. We started from initial values identical to those in Xu et al. (2017).

Both the learning rate and dropout followed a linear decay schedule in which they would have a final value of 0.0064 and 0.1024, respectively, by the end of training. We also conducted tests that showed that having a constant, lower dropout rate, yielded slightly better results. We did not use them for the EDL task due to time constraints.

5.6 Candidate Pruning

We exhaustively generated all the possible mention candidates that we passed to the classification step. The output is a probability distribution for each class (named and nominal), whose sum is 1. We used the highest probability class to tag the mention if it was 0.5 or greater, otherwise we ignored the output and assigned it to the `NONE` class.

No overlapping mentions were output, instead each mention had to have no overlap at all. We evaluated two different algorithms to determine which mentions to keep: The highest probability and the longest match:

- The highest probability algorithm proceeds from left to right and uses the highest probability, nonoverlapping, leftmost mention.
- The longest match instead uses the longest, nonoverlapping, rightmost mention.

During testing, the highest probability algorithm produced the best results, a few points greater than the longest first. The output was also visually cleaner upon manual inspection. We did a grid search for the cutoff value and found that 0.5 produced the best results. Nonetheless, both 0.4 and 0.6 yielded similar results and would be reasonable choices as well.

6 Entity Linking

6.1 Generation of Entity Candidates

We used the mentions from the recognition step to produce the entity candidates. Each mention found by the FST dictionary has a unique ID that serves as entry point to the mention-entity graph (Sect. 3.3).

6.2 Construction of a Local Graph to Disambiguate Entities

We build a local graph to disambiguate the entities in a text. The nodes of the graph consist of the mentions and the candidates. We link these nodes with three types of edges:

1. The mention-to-candidate edges;
2. The entity cooccurrence edges linking entities when they cooccur in the Wikipedia corpus;

Language	Named			Nominal			Overall		
	P	R	F1	P	R	F1	P	R	F1
English	0.734	0.816	0.773	0.580	0.805	0.674	0.801	0.676	0.733
Chinese	0.769	0.792	0.780	0.554	0.757	0.639	0.769	0.612	0.682
Spanish	0.736	0.685	0.709	0.584	0.657	0.618	0.736	0.567	0.640

Table 2: Results from evaluating on the 2016 data (not including wikipedia dictionary).

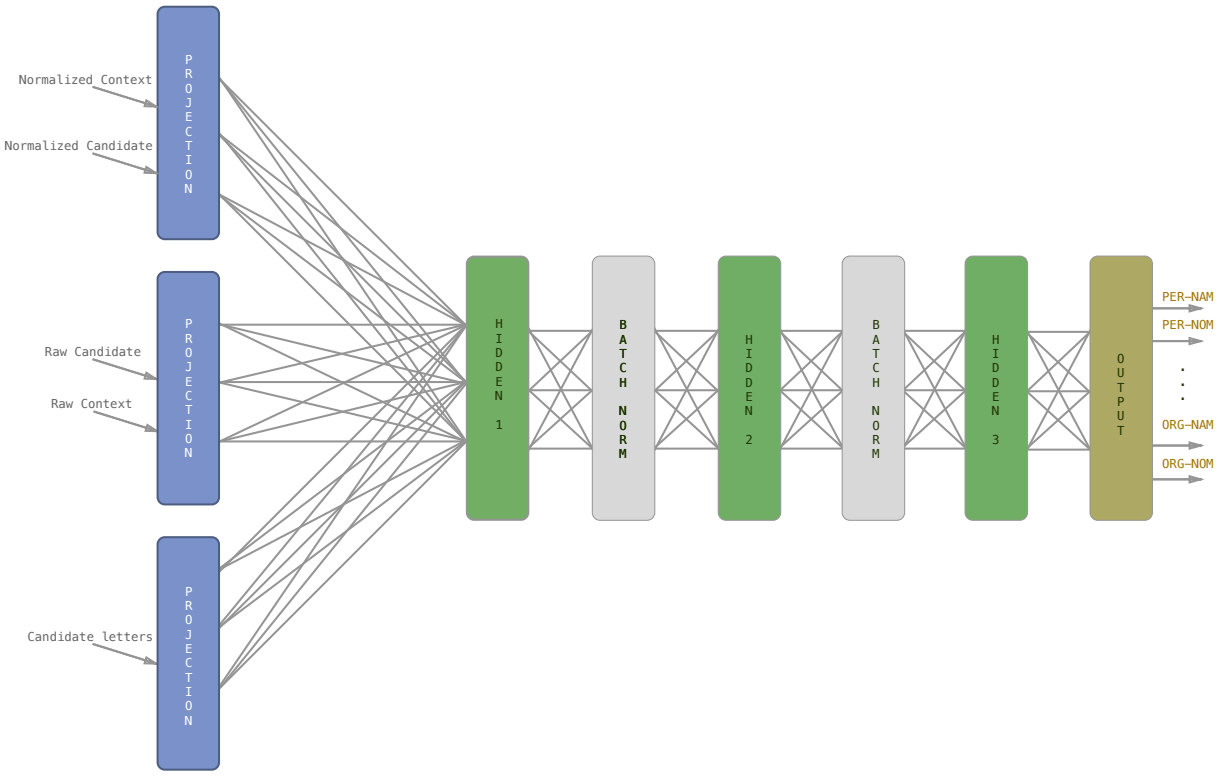

Figure 3: Overview of the full NER network architecture

3. The entity inlink edges, reflecting links between pages (entities) in Wikipedia;

While the mentions are language-dependent, the entities reside in a multilingual domain and their edges are aggregated from all the editions of Wikipedia we consider. Fig. 4 shows an overview of the graph.

Following Södergren and Nugues (2017), the graph is weighted using the PageRank algorithm (Brin and Page, 1998). The candidates are ranked per mention and a normalized weight is produced. The top three candidates are reordered if the candidate title exactly matches the mention.

In contrast to Södergren and Nugues (2017), we included the inlinks and we carried out the disambiguation inside a window of 20 mentions. We introduced this window method to reduce the computation and the upper-bound execution time. At the start of the disambiguation, the window is set at the beginning of the text and then shifted by 10 mentions. The windows are then partially overlapping and, in case of conflict, we use the rankings from the left one.

6.3 Reranker

To reduce errors made by the disambiguator and introduce a NIL candidate, we trained a reranker on the TAC 2015-2016 data. We generated a training set of examples by applying the graph-based

Name	Value
Weight initialization	RELU Uniform
Max. window size	7
Epoch count	160
Learning rate	0.1024
Dropout	0.4096
Optimizer	ADAM
L2 regularization	0.0
Neuron count	512
Batch size	512
Activation function	Leaky RELU

Table 3: The final hyperparameters used in the TAC2017 EDL task

disambiguator to all the available annotated text. We limited the set of candidates to the top three entities for each mention or up to the correct one if necessary. We then marked each candidate in these lists as positive or negative according to the gold standard. We assigned all the detected mentions overlapping gold standard mentions to the NIL entity. We discarded the rest.

We used two sets of features, *candidate* and *context*, resulting in two models:

1. The candidate set contains the Jaccard similarity coefficient between the entity title and the mention, the PageRank weight, and the commonness defined as $P(E_q|M_i)$, where E is the entity and M is the specific mention. All the features in the candidate set are encoded as a quantitized one-hot encoded array.

2. The context set includes the *candidate* features and additional FOFE-encoded left and right contexts surrounding the mention using the same word embeddings as the NER derived from Wikipedia.

We trained the reranker as a binary classifier using a feed-forward neural network with binary cross entropy loss and sigmoid activation. The network consists of 3 dense layers of size 64 for the *candidate* model and 128 for the *context* model.

We incorporated the *candidate* and *context* models into the entity disambiguation using the

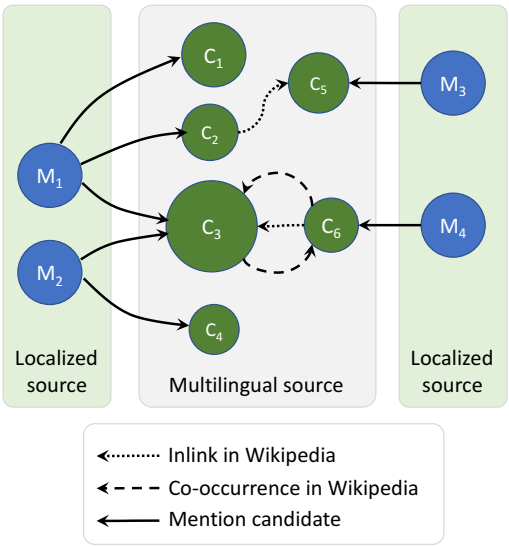

Figure 4: Overview of the entity disambiguation graph

following equation to produce final ranking score:

$$\text{Final score} = RV \cdot RRS^{\alpha} \quad (3)$$

where RV is rank value and RRS is the rerank score, which is equivalent to the prediction probability.

We performed a grid search to find the optimal α for the reranker using the gold standard training set and we selected the best of the two models for each language.

6.4 Postprocessing

The postprocessing stage consists of the following steps: a baseline coreference resolution, a nominal discovery, and a filtering. The baseline coreferencing method uses the linked mentions as input and tries to find exact matches of the first and last word of each linked mention in the text. When such a match is found, the two mentions are coreferring.

To discover the nominal mentions, we first collected a seed word set from the nominal mentions in the TAC data. We then built a dictionary, where the keys were the entities and the values, the nominal phrases compatible with each entity. We extracted these phrases from the Wikidata description of the corresponding entity, as well as the labels and aliases of its instances-of and occupation

Language	en	es	zh
NERC	U_2: 0.833	U_2: 0.804	U_1/U_3: 0.760
NEL	U_4: 0.751	U_2: 0.751	U_4: 0.718
NELC	U_4: 0.726	U_2: 0.733	U_1/U_3: 0.760
CEAFm	U_4: 0.783	U_3: 0.728	U_4: 0.736

Table 4: Best version for the named class. F1 score. Results on the TAC 2017 data

Languages	en					es					zh				
	U_1	U_2	U_3	U_4	U_5	U_1	U_2	U_3	U_4	U_5	U_1	U_2	U_3	U_4	U_5
NERC	0.813	**0.833**	0.825	0.813	0.797	0.802	**0.804**	0.788	0.749	0.763	**0.760**	0.750	**0.760**	0.750	0.741
NEL	0.732	0.738	0.730	**0.751**	0.691	0.747	**0.751**	0.746	0.701	0.687	0.716	0.691	0.716	**0.718**	0.690
NELC	0.711	0.717	0.710	**0.726**	0.668	0.784	**0.788**	0.788	0.745	0.765	**0.757**	0.755	**0.757**	0.746	0.741
CEAFm	0.752	0.751	0.752	**0.783**	0.754	0.722	0.714	**0.728**	0.685	0.700	0.726	0.702	0.726	**0.736**	0.720

Table 5: Breakdown of results for runs U_1 to U_5, F1 scores, bold indicates the best result per language

relations. We finally intersected the resulting set with the seed word set.

Finally, we filtered out the mentions, where we could not find an acceptable class using the entity-to-class mapping dictionary or using the NER predicted class, when available.

7 Results

Ugglan was primarily targeting the named entity disambiguation. It was not designed for the nominal and NIL entities, and hence its results on these categories are not at the same level in terms of accuracy. Therefore, we will merely analyze the named results, where the result categories correspond to these acronyms:

NERC, Named Entity Recognition and Classification, corresponding to `strong_typed_mention_match` in the evaluation script.

NEL, Named Entity Linking (`strong_link_match`);

NELC, Named Entity Linking and Classification (`strong_typed_link_match`);

CEAFm, Clustered Mention Identification (`CEAFm`).

We submitted five runs, U_1 to U_5. Table 4 shows an overview of the best combination per language and type of result taken from the official evaluation data and Table 5 shows the full breakdown.

The pipeline setup for the particular runs were selected using the TAC 2016 evaluation as a guide. The runs U_1 to U_3 used available training data from TAC 2014-2016, while U_4 and U_5 only used TAC 2014-2015. Table 6 shows the different configurations.

From the results in Table 4, the typed classification is best using only the FOFE-based NER. The Stanford NER is better when it comes to clustered mentions.

8 Discussion

8.1 Mention Recognition

Ugglan's ability to find linkable mentions is determined by the recall level of the FST dictionary. The NER only helps in reducing noise, thus increasing precision at the expense of possibly lowering the overall recall. The *hybrid* mode tries to mitigate the recall loss by including mentions which have no overlap with any NER mention. NIL mentions are only found using a NER or if the mention was linked and could not be resolved to a Freebase entity. The FOFE NER was trained to identify NOMs, but these were never used as they could not reliably be linked to existing linkable mentions.

8.2 Linking

The windowing approach limits the computation complexity at the expense of a possible lower precision. We did not evaluate the effects of the window size and the values were picked arbitrarily

	Mention Recognition						NER			Reranker								
	NER-only			Hybrid						Candidate			Context			None		
	en	es	zh	en	es	zh	en	es	zh	en	es	zh	en	es	zh	en	es	zh
U_1	✓	✓				✓	F	F	F				✓	✓	✓			
U_2	✓	✓	✓				F	F	F				✓	✓	✓			
U_3				✓	✓	✓	F	F	F	✓				✓	✓			
U_4				✓	✓	✓	S	S	F*	✓				✓	✓			
U_5				✓	✓	✓	S	F*	S							✓	✓	✓

Table 6: System configuration. F stands for FOFE and S for Stanford NER. F* is an older FOFE model

using human insight only. Arguably, the optimal size depends on the impact of topic drift, as the linker performs best with a coherent and compatible context with as many related mentions as possible. The more diverse the context is in terms of mentions and candidates, the noisier the graph becomes and the relevant context may shrink as it would require a bigger context to get sufficient supporting candidates to produce a good linkage.

Acknowledgments

This research was supported by Vetenskapsrådet, the Swedish research council, under the *Det digitaliserade samhället* program.

References

Sergey Brin and Lawrence Page. 1998. The anatomy of a large-scale hypertextual web search engine. *Computer Networks* 30(1–7):107–117. Proceedings of WWW7.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics (ACL 2005)*. Ann Arbor, pages 363–370.

Heng Ji and Joel Nothman. 2016. Overview of TAC-KBP2016 trilingual entity discovery and linking and its impact on end-to-end cold-start kbp. In *Proceedings of the Ninth Text Analysis Conference (TAC2016)*.

Heng Ji, Joel Nothman, and Ben Hachey. 2014. Overview of TAC-KBP2014 trilingual entity discovery and linking. In *Proceedings of the Seventh Text Analysis Conference (TAC2014)*.

Heng Ji, Joel Nothman, Ben Hachey, and Radu Florian. 2015. Overview of TAC-KBP2015 trilingual entity discovery and linking. In *Proceedings of the Eighth Text Analysis Conference (TAC2015)*.

Heng Ji, Xiaoman Pan, Boliang Zhang, Joel Nothman, James Mayfield, Paul McNamee, and Cash Costello. 2017. Overview of TAC-KBP2017 13 languages entity discovery and linking. In *Proceedings of the Tenth Text Analysis Conference (TAC2017)*.

Marcus Klang and Pierre Nugues. 2017. Docforia: A multilayer document model. In *Proceedings of the 21st Nordic Conference of Computational Linguistics*. Gothenburg, page 226–230.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Atlanta, Georgia, pages 746–751. http://www.aclweb.org/anthology/N13-1090.

Anton Södergren and Pierre Nugues. 2017. A multilingual entity linker using PageRank and semantic graphs. In *Proceedings of the 21st Nordic Conference of Computational Linguistics*. Gothenburg, page 87–95.

Mingbin Xu, Hui Jiang, and Sedtawut Watcharawittayakul. 2017. A local detection approach for named entity recognition and mention detection. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 1237–1247.

ShiLiang Zhang, Hui Jiang, MingBin Xu, JunFeng Hou, and LiRong Dai. 2015. The fixed-size ordinally-forgetting encoding method for neural network language models. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Association for Computational Linguistics, Beijing, China, pages 495–500. http://www.aclweb.org/anthology/P15-2081.