UNIVERSITÀ DEGLI STUDI DI TRENTO

CIMeC - Center for Mind/Brain Sciences

Master's Degree in Cognitive Science

Academic Year
2016 – 2017

Exploiting a knowledge-based probabilistic model for optimally combining alternative NERC and Entity Linking candidate annotations on the same textual mention

Supervisor:
Prof. Paolo Bouquet
Dr. Marco Rospocher

Co-supervisor:
Dr. Francesco Corcoglioniti
Prof. Claire Gardent

Student:

Yiqing Liang

Abstract

Knowledge extraction is typically the result of the combination of different NLP techniques, implemented in different tools, extracting various kind of information: e.g., maned entities, semantic frames, temporal expressions, entity types. As this information is independently derived, contradicting content may be produced from the same piece of text. Hence, it is crucial to research and develop techniques to coherently select/harmonize/complement the content returned by the various NLP tools used, in order to improve the overall quality of the knowledge extracted from text. In this research project we investigate a probabilistic model, derived from the ontological knowledge extracted from existing knowledge bases (Yago and DBpedia) and an annotated corpus (AIDA-CONLL), to maximize the joint probability of NERC and entity linking candidate annotations on the same textual mention. After being tested on gold-standard, our model was found to improve significantly the performance of Entity Linking significantly, while the improvement for NERC is relatively not obvious. This project is carried on within the context of a state-of-the-art knowledge extraction framework: PIKES.

Contents

1 Introduction

1.1 Problem to be Solved?

Knowledge extraction from text involves techniques from different NLP fields, thus, in a knowledge extraction system, such different techniques are usually implemented by different tools developed independently. When a piece of text is processed by the system, various kinds of information is produced through pipelines formed by different tools, between which no connection is made until their tasks are finished. As a result, contradictions may occur between annotations produced by different tools for the same text.

When there exists a contradiction, it's obvious that at least one of the annotations is incorrect.

This project is mainly focused on extracting coherent and compatible knowledge of two sorts of annotations, Named Entity Recognition and Classification (NERC) and Entity Linking (EL), which are two basic and very important NLP tasks almost always involved in a knowledge extraction pipeline.

Here is an example of incoherency between NERC and Entity Linking. For the mention "Liverpool" in the sentence "Paul and John started playing the guitar in Liverpool", the tool for NERC returns the type "ORGANIZATION", while the tool for Entity Linking return the entity "Liverpool", which is a city, a place. A place can't be an organization, so here goes the contradiction. We can easily understand that the Entity Linking result here is correct but the NERC is not. "Liverpool" is a location instead of an organization, so the NERC type should be "LOCATION".

Therefore, by researching and developing a technique to harmonize the content returned by various NLP tools, it helps not only to extract coherent knowledge but also to increase the overall accuracy of the whole system.

1.2 Related Work

There are in principle two ways of tackling the problem of producing consistent annotations: (1) apply tools for NER and EL separately and a-posteriori compare and fix their outcomes, which generally requires those tools to return not just the best annotations but also all the candidate annotation values with their scores, and (2) build a unique tool that *jointly* perform NER and EL, that is explicitly designed to exploit the synergies among the two tasks.

Systems that are trained in the latter way for Named Entity Recognition and Entity Linking have been researched on before.

Stern et al. [15] proposed a model relying on combined NER modules which transfer the disambiguation step to the EL component, where referential knowledge about entities can be used to select a correct entity reading. Hybridation is a main feature of the system, as they have performed experiments combining two types of NER, based respectively on symbolic and statistical techniques. Furthermore, the statistical EL module relies on entity knowledge acquired over a large news corpus using a simple rule-base disambiguation tool.

A system introduced by Guo et al. [7] jointly optimizes mention detection and entity disambiguation as a single end-to-end task. This system's main focus is on extracting Entity Linking information from microblog posts on Twitter. They proposed a structural SVM algorithm, combining structural learning and a variety of first-order, second-order, and context-sensitive features. The system JERL proposed by Luo et al. [12] also have the similar idea but with a different implementation. This system was designed for general text instead of specially for microblogs. It jointly model NER and linking tasks and capture the mutual dependency between them.

Another model called NereL [14] takes a large set of candidate mentions from typical NER systems and a large set of candidate entity links from EL systems, and ranks the candidate mention-entity pairs together to make joint predictions.

All the methods above intend to take advantage of the dependency between NER and EL and benefit one from another during the training stage, and none of them exploit the use of NERC type. In other words, they all take the (2) way above, and the (1) way has seldom been explored. However, in our model, we intend to take the (1) way, in which the process for producing NERC and EL are completely independent, and the results are optimized afterward.

We chose this method for the motive that, there are many state-of-the-art tools for NERC and Entity Linking (separately) and we want to exploit them and all the knowledge accumulated so far in their realization, rather than building from scratch our own NERC+EL tool that we would have to improve and maintain ourselves with a sensible effort.

1.3 Proposed Solution

Instead of training a model that merges the NERC and EL process, this thesis investigates a probabilistic model exploiting background knowledge in a knowledge base, under which the NERC and EL candidates with a maximum joint probability are selected and taken as the final annotation after they are independently produced.

Under a same text mention, a few candidate annotations for NERC and EL are produced by different tools respectively, all with a confidence score, indicating their probability as the correct annotation for this mention. With the ontological knowledge of the linked entities extracted from existing knowledge bases, (i.e., the type information of the corresponding entity in the knowledge base), we calculate the joint probability of all the possible pairs of NERC and EL candidates along with the entity types of the linked entity candidate. We choose the NERC/EL pair with the maximum joint probability as the ultimate result.

1.4 Framework

This project is carried on within the context of a state-of-the-art knowledge extraction framework: PIKES [3].

PIKES is a Java-based suite that extracts knowledge from textual resources. The tool implements a rule-based strategy that reinterprets the output of semantic role labelling (SRL) tools in light of other linguistic analyses, such as dependency parsing or co-reference resolution, thus properly capturing and formalizing in RDF important linguistic aspects such as argument nominalization, frame-frame relations, and group entities.

In PIKES, tasks of NERC and Entity Linking are both carried out to detect entities in a text and resolve them against DBpedia (the ones not linked to DBpedia becomes new entities with their own identifiers). PIKES uses the NERC tool from Stanford CoreNLP [6], and the DBpedia Spotlight EL tool [1].

1.5 Outline of the Thesis

In chapter 2 we review some background knowledge of techniques and datasets involved in the project. Chapter 3 includes the detailed introduction and mathematical derivation of out model and how it was implemented. Chapter 4 reports on the evaluation on the model and its performances in improving the outputs of NERC and Entity Linking. Chapter 5 summarizes all the work we've done and provides further discussion.

2 Background and State of Art

This chapter introduces several tool and techniques that are involved in the project, including two NLP tools, two knowledge bases, some datasets and scripts.

2.1 NLP tasks

2.1.1 NERC

Named Entity Recognition and Classification (NERC) is a task that locates the named entities in text and classifies them into one of the few pre-defined categories, including but not limited to person, organization, location, date, time, ordinal and misc.

Taking the example mentioned in Chapter 1 for explanation: in the sentence "Paul and John started playing the guitar in Liverpool", the named entities "Paul", "John" and "Liverpool" should be located, and the three entities should also be classified as "PERSON", "PERSON" and "LOCATION".

For the NERC task, here we use the tool Stanford Named Entity Recognizer (NER) [6]. Implemented in Java, Standford NER labels sequences of words in a text which are the names of things, such as person and company names, or gene and protein names. The software is based on the use of a CRF that finds the best labeling of a sequence of tokens (i.e., a sentence), allowing exploiting the dependencies between tokens at different positions in a sequence (e.g., the fact that locations are typically introduced by certain prepositions, etc...).

Included with Stanford NER are three models trained on different datasets, with different classes defined. There's a 4 class model trained on CoNLL 2003 eng.train data set, a 7 class model trained on the MUC 6 and MUC 7 [2] training data sets, and a 3 class model trained on both data sets and some additional data. The classes defined for each model is as follows:

- 3 class: Location, Person, Organization
- 4 class: Location, Person, Organization, Misc
- 7 class: Location, Person, Organization, Money, Percent, Date, Time

Among the three models, we adopt the second one with 4 classes because the extra Percent, Date, Time classes have no corresponding entities in the knowledge bases, and thus they cannot be linked by EL, meaning that our approach cannot fix errors in their detection. In other words, we cannot do anything about them and thus we ignore them.

Given a piece of text, the software returns sequence labeling that may need further processing for our needs.

2.1.2 Entity Linking

Entity Linking is the task of determining the identity of entities mentioned in text. It requires a knowledge base containing the entities to which entity

mentions can be linked.

For example, given an entity "Liverpool", the entity linking system has to decide if this entity corresponds to the city Liverpool or the football club Liverpool FC or some other entities in the knowledge base, depending on the surface form and the context. Usually, named entities spotting is also part of the entity linking task. The difference between NERC and entity linking is that, after recognizing a named entity from text, NERC returns its category, while entity linking tells which entity in knowledge base it corresponds to.

The tool we use for entity linking is DBpedia Spotlight[5]. It's is a tool for automatically annotating mentions of DBpedia resources in text.

The implementation of DBpedia Spotlight consists of two parts, phrase spotting and disambiguation.

Phrase spotting is for finding phrases in a text that later should be linked to DBpedia entities. It also consists of two steps.

In the first step, candidates for possible annotations are generated. There are two implementations for this step, language-independent implementation and language-dependent implementation. In the language-independent one, the candidates are generated by traversing a finite state automaton encoding all possible sequences of tokens that form known spot candidates; in the language-dependent one, sequences of capitalized tokens, noun phrases, prepositional phrases and multi word units, named entities are all identified as candidates.

In the second step, the best candidates from the set of phrases generated in first step are selected. For each spot candidate, a score based on a combination of features are computed. The overlapping candidates with lower score and candidates with a score lower than a specific score threshold are removed.

Disambiguation is for finding entities from DBpedia linked to phrases found in phrase spotting step. The algorithm is based on a generative probabilistic model from [8].

The score for an entity e given the phrase s and context c is calculated as a combination of $P(e)$, $P(s|e)$ and $P(c|e)$. The combination could be either a linear regression mixture or the product of the individual values. For this project, we employ the latter one, as it's the default setting of the software and it's more mathematically explainable.

Hence, under a specific mention m, for computing the joint probability of $P(m, e)$, meaning that a piece of text is a mention m and that text refers to a DBpedia entity e, we have:

$$P(m, e) = P(e, s, c) = P(e)P(s|e)P(c|e) \tag{1}$$

where $P(e)$ corresponds to the popularity knowledge (the prior probability of an entity appearing in a document), $P(s|e)$ corresponds to the name knowledge (the likelihood of a name referring to a specific entity) and $P(c|e)$ corresponds to the context knowledge (the likelihood of an entity appearing in a specific context).

In this generative model, probabilities are negligibly small and are represented in logarithmic space to avoid underflows. To produce a more useful

score, for each entity, a softmax function was applied to normalize the score and to obtain a final disambiguation score between 0.0 and 1.0.

The disambiguation scores of all the candidates for a certain mention always add up to 1, thus, it can be viewed as a probability that a certain candidate entity should be linked to the mention, written as $P(e|m)$.

2.2 Background Knowledge Resources

2.2.1 YAGO

YAGO is the knowledge base from which we extract the ontological knowledge and type information for later probability calculation.

YAGO[16] is an open source knowledge base developed at the Max Planck Institute for Computer Science in Saarbrücken. It's built on entities and relations automatically extracted From Wikipedia and unified with WordNet[13].

The formalism YAGO is based on is a slight extension of RDFS, the *YAGO model*. It can express relations between facts and relations, while it is at the same time simple and decidable.

In YAGO model, all objects are represented as *entities*. Two entities can stand in a *relation*. For example, the entity *Albert Einstein*, it stands the HasAcademicAdvisor relation with the entity *Alfred Kleiner*, meaning that Albert Einstein had an academic advisor named Alfred Kleiner, written as:

Albert Einstein hasAcademicAdvisor *Alfred Kleiner*

Numbers, dates, strings and other literals are represented as entities as well. This means that they can stand in relations to other entities, but only in the object position, i.e., as values.

In the YAGO model, words are entities as well. This makes it possible to express that a certain word refers to a certain entity.

However, the information most useful for our project is the *class* information.

In knowledge base, similar entities are grouped into *classes*. An entity could belong to multiple classes, and classes could also comprise many entities. Each entity has to be an *instance* for at least one class. The relation that connects entities and the classes they belong to is TYPE relation. For example, *Albert Einstein* is an instance of class *physicist*, it's expressed as:

Albert Einstein TYPE *physicist*

About the classes, it should be noticed that:

- Classes are also entities
- Classes are arranged in a taxonomic hierarchy, expressed by the subClassOf relation
- The TYPE relations are extracted from the category system of Wikipedia, while the subClassOf relations are extracted from WordNet
- Only leave categories from Wikipedia are extracted, thus there's no hierarchy among them

- Each synset of WordNet becomes a class of YAGO, among which there is hierarchy
- Leave categories from Wikipedia are later mapped as subclasses to higher WordNet classes

Still, let's take the entity *Albert Einstein* as an example to see how the class hierarchy is organized.

The classes *agnostics* and *Swiss physicists* are two of the many Wikipedia categories that entity *Albert Einstein* belongs to. Class *agnostics* is subclass of two WordNet classes, *person* and *agnostic*, while *agnostic* is also a subclass of *person*. In the meantime, *Swiss physicists* leads to two WordNet classes, *person* and *physicist*, above *physicist* we get *scientist*, which also has a superclass as *person*.

Therefore, all the classes we mention before, including *agnostics*, *Swiss physicists*, *agnostic*, *physicist*, *scientist* and *person* are classes in YAGO that entity *Albert Einstein* belongs to. There are also classes above *person*, but they're too general to give much useful information, so it's not discussed in this example.

Later, in our model, all the classes that an entity belongs to would be treated as a set for further calculation.

2.2.2 DBpedia

We apply the tool DBpedia Spotlight for Entity Linking task, and this tool links the mentions to specific entities in DBpedia knowledge base[1].

Similar to YAGO database, DBpedia also aims to extract structured data from Wikipedia.

Entities of DBpedia are also originally pages of Wikipedia. For example, the entity *Barack_Obama*, represented in DBpedia URI as *http://dbpedia.org/page/Barack_Obama*, comes from the Wikipedia page of *https://en.wikipedia.org/wiki/Barack_Obama*.

Due to the fact that YAGO and DBpedia have the same resource (each entity corresponds to one Wikipedia page), they have the same set of entities, i.e., a DBpedia entity could always be found with the same name of the one in YAGO, and vice versa.

Although DBpedia and YAGO are extracted the same resource, they still have a lot of differences as listed:

- YAGO extracts only 14 relationship types, such as *subClassOf*, *locatedIn* from sources like category system and Wikipedia redirects, while DBpedia extracts more types of relationship from MediaWiki and artical texts
- DBpedia performs an infobox extraction but YAGO does not
- For determining (sub-)class relationships, YAGO does not use the full Wikipedia category hierarchy, but links leaf categories to the WordNet hierarchy, while DBpedia uses all Wikipedia and YAGO categories.

In other words, for finding the categories that an entity belongs to, DBpedia could provide not only all the entity's YAGO categories, but also more categories extracted using Wikidata, SKOS Vocabulary, FOAF, etc.

2.3 Annotated Datasets

2.3.1 CONLL-AIDA2013

We take the CoNLL-AIDA2013 annotated dataset as our training data and gold-standard for evaluating the performance of our approach.

The CoNLL-AIDA data is a large whole-document named entity linking dataset whose source data is from the Reuters news corpora [11]. The dataset consists of 1393 documents. There are two layers of manual annotations for this dataset, one is for NERC and another is for Entity Linking. These two parts are annotated independently by different people in different formats.

Tjong Kim Sang & De Meulder [17] tokenised and manually annotated the documents with PER, ORG, LOC and MISC entity mentions, which are exactly the gold-standard we need for NERC type. The format of this file is shown in Figure 1. Each document is started with a line as "-DOCSTART- -X- O O", and each line after that represents one token in the file. The columns for each line are organized as:

1. Token
2. Part-of-Speech tag
3. Phrase structure
4. NERC type, "O" if the token is not part of a named entity

```
-DOCSTART- -X- 0 0

EU NNP I-NP I-ORG
rejects VBZ I-VP 0
German JJ I-NP I-MISC
call NN I-NP 0
to TO I-VP 0
boycott VB I-VP 0
British JJ I-NP I-MISC
lamb NN I-NP 0
. . 0 0
```

Figure 1: Format of NERC gold-standard

Hoffart et al. [9] annotated all named entity mentions with YAGO2 entity annotations, or NIL if there is no corresponding KB entity. Figure 2 shows an example of this Entity Linking gold-standard file. The beginnings of each ducument are marked with a line as "-DOCSTART- (some_doc_id)" where the

"some_doc_id" consists of the document number and its first token. The columns of a line represents:

1. Token
2. Mention span: "B" for mention begin, "I" for inside mention, empty column for outside
3. Mention text: this is a bit redundant, but a sanity check when reading the output
4. Entity identifier: where a mention is linked to the KB, this will be the id/title (e.g., a Wikipedia title). Where the mention is a NIL, this column should be blank

```
-DOCSTART- (1 EU)
EU      B       EU
rejects
German  B       German  Germany
call
to
boycott
British B       British United_Kingdom
lamb
.

Peter   B       Peter Blackburn
Blackburn       I       Peter Blackburn
```

Figure 2: Format of Entity Linking gold-standard

2.3.2 Evaluation scripts

The evaluation is separated into two part, NERC and Entity Linking, with two sets of evaluation scripts and different evaluation methods.

There are four measures of evaluation used in the scripts, accuracy, precision, recall and F_1 score (also known as FB1 or F-score in the result returned by the evaluation scripts). They are computed as follows (In the formulas, TP = True Positive, TN = True Negative, FP = False Positive, FN = False Negative):

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2}$$

$$Precision = \frac{TP}{TP + FP} \tag{3}$$

$$Recall = \frac{TP}{TP + FN} \tag{4}$$

$$F_1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{5}$$

The evaluation of the NERC systems is usually done following the CoNLL 2002 and CoNLL 2003 shared tasks methodology. Those tasks also distribute an official evaluation script which we will be using to evaluate the performance of the NERC types in our system.

Before evaluation, a gold-standard corpus in CoNLL 2003 format (same as the example shown in Figure 1) must be provided, and the NERC annotations produced by the model also has to be converted into the same format.

The result returned by the evaluation scripts is like Figure 3. The first line states the number of tokens in file, number of tokens which are part of a mention, total number of mentions and number of correct NERC types. The line below tells the overall accuracy of the whole model, followed by precision, recall and F_1 measures. The following four lines are evaluations for respective NERC types.

```
processed 302811 tokens with 34835 phrases; found: 33862 phrases; correct: 32088.
accuracy:  98.32%; precision:  94.76%; recall:  92.11%; FB1:  93.42
             LOC: precision:  97.27%; recall:  94.56%; FB1:  95.89  10314
            MISC: precision:  92.80%; recall:  88.18%; FB1:  90.43  4725
             ORG: precision:  94.37%; recall:  91.99%; FB1:  93.16  9025
             PER: precision:  93.43%; recall:  91.59%; FB1:  92.50  9798
```

Figure 3: Example of NERC evaluation result

The evaluation of Entity Linking is described by Cornolti et al. [4]. This includes evaluation measures for mention detection, end-to-end linking and document tagging. A number of measures are adopted in the evaluation script.

tp	fp	fn	precis	recall	fscore	match
26425	21910	8504	0.547	0.757	0.635	strong_mention_match
22445	25890	5372	0.464	0.807	0.589	strong_linked_mention_match
16389	31946	11428	0.339	0.589	0.430	strong_link_match
3769	6930	1826	0.352	0.674	0.463	entity_match

Figure 4: Example of Entity Linking evaluation result

As shown in the example of Figure 4, the evaluation result contains four kinds of match, each of them has six scores. From left to right, they are: number of True Positive, False Positive, False Negative, Precision, Recall and F-score. The explanation of match measures is:

- Mention detection evaluation
  - weak mention match (not used in the evaluation result or in Figure 4): A micro-averaged evaluation of entity mentions. The system mention extent must have at least one token in common with the aligned gold mention

  - strong mention match: The same as *weak mention match* but is stricter, requiring the system and gold mentions to be exactly the same.
  - strong linked mention match: Similar to *strong mention match* but only mentions linked to an entity are taken into account

- End-to-end linking evaluation
  - entity match: A micro-averaged evaluation of links. The system mention extent must have at least one token in common with the aligned gold mention and the link must be to the same KB title
  - strong link match: The same as *entity match* but is stricter, requiring the system and gold mentions to be exactly the same

The two measures for mention detection evaluation do not depend on the entity that has been linked to the mention. As our model only changes the annotations of a mention but not the mention extent itself, the mention detection evaluation part should always be the same before and after the model, so we're mostly interested in the end-to-end linking evaluation part.

3 Approach

3.1 Intuition

As introduced in Chapter 1, this project aims to exploit a probabilistic model to maximize the joint probability of NERC and Entity Linking candidates. In this section, we will provide more details behind the idea in an intuitive way.

Given some text for annotation, PIKES returns a NAF (NLP annotations) file containing various kinds of language annotations, among which there are NERC and Entity Linking candidates for each mention recognized as named entity. Figure 5 shows an example of how this information is presented in NAF file. Among the external references, the ones with attribute *resources="value-confidence"* are NERC types and the ones with *resource="dbpedia-candidates"* are URIs of linked entities from DBpedia knowledge base.

It can be seen from the figure that, for each entity, there are several NERC and Entity Linking candidates, each candidate with a confidence score, indicating the probability that this candidate is the correct one. For both NERC and Entity Linking, the candidate with the highest score is considered the most "probable" candidate, or the "correct" NERC type or linked entity for the time being.

However, the system is not perfect and sometimes the so far best candidates could be actually wrong. An obvious sign is that the NERC type and Entity Linking candidates don't seem to be "compatible". Take the entity in Figure 5 as an example. The mention "Canada" has the best NERC type as *LOCATION* and Entity Linking candidates as *Canada_national_rugby_union_team* (Here we only consider the entity name and ignore the long prefix in front). Under the context, "Canada" here should mean the country. The incompatibility between these two annotations is very explicit, so is the fact that the Entity Linking candidate *Canada_national_rugby_union_team* is wrong.

Judging from the example above and also the common sense we have, we can easily conclude that, if there exists this kind of incompatibility, or contradiction, between NERC and Entity Linking annotations, at least one of these annotations should be incorrect. We can take advantage of this phenomenon and find a way to eliminate the incompatibility, thereby also change the wrong annotation to a correct one.

Let's assume an example in which the correct NERC type and Entity Linking candidate for a certain mention exist in the NAF file and are among the external references, but one of them is not the one with the highest confidence score. Let's also assume that the best candidates of the two annotations have contradiction between them.

If we had a way to measure the "compatibility score" between the two annotations, we would be able to find that the compatibility between them is quite low, and then we could fix it by selecting the ones more compatible to each other among other candidates, which have a much higher probability to be correct.

In order to do that, we take advantage of the knowledge bases and the

```
<entity id="e15" type="LOCATION">
  <references>
    <!--Canada-->
    <span>
      <target id="t44" />
    </span>
  </references>
  <externalReferences>
    <externalRef resource="value-confidence" reference="LOCATION" confidence="0.9999945" />
    <externalRef resource="value-confidence" reference="ORGANIZATION" confidence="2.9206583E-6" />
    <externalRef resource="value-confidence" reference="PERSON" confidence="4.964267E-7" />
    <externalRef resource="value-confidence" reference="MISC" confidence="2.0568157E-6" />
    <externalRef resource="dbpedia-candidates" reference="http://dbpedia.org/resource/Canada_women's_national_ice_hockey_team" confidence="4.947758E-13" />
    <externalRef resource="dbpedia-candidates" reference="http://dbpedia.org/resource/Canada_men's_national_soccer_team" confidence="0.117250875" />
    <externalRef resource="dbpedia-candidates" reference="http://dbpedia.org/resource/Canada_men's_national_junior_ice_hockey_team" confidence="1.4696172E-14" />
    <externalRef resource="dbpedia-candidates" reference="http://dbpedia.org/resource/Canada_(New_France)" confidence="1.5606653E-9" />
    <externalRef resource="dbpedia-candidates" reference="http://dbpedia.org/resource/Canada_men's_national_ice_hockey_team" confidence="0.00232604" />
    <externalRef resource="dbpedia-candidates" reference="http://dbpedia.org/resource/Canada_national_rugby_union_team" confidence="0.7578514" />
    <externalRef resource="dbpedia-candidates" reference="http://dbpedia.org/resource/Canada_women's_national_soccer_team" confidence="9.7909114E-15" />
    <externalRef resource="dbpedia-candidates" reference="http://dbpedia.org/resource/Canada" confidence="0.1225715" />
    <externalRef resource="dbpedia-candidates" reference="http://dbpedia.org/resource/Province_of_Canada" confidence="2.3510902E-10" />
    <externalRef resource="dbpedia-candidates" reference="http://dbpedia.org/resource/Canada_national_cricket_team" confidence="1.7453073E-7" />
  </externalReferences>
</entity>
```

Figure 5: Information representation in NAF

annotated datasets mentioned in the last chapter.

In the annotated datasets, we have the correct NERC type and Entity Linking for all the named entities in corpora. Thus, we could extract the information on concurrence of specific NERC type and Entity Linking. For example, if a certain mention has the NERC type as *LOCATION* and is linked to an entity *Washington,_D.C.* in knowledge base, we know that this NERC type and this linked entity are compatible, because they appear under the same mention. It can also be deduced that, the more times a NERC type and a linked entity appear together, the more compatible these two are.

However, the annotated dataset we have is of limited size, and it does not contain all the possible combinations of NERC type and linked entity. Therefore, it's necessary to find a way to generalize this method.

There exists only four possible NERC types, but millions of possible entities, so the main problem is to reduce the number of entities. Instead of counting directly the concurrence number of a specific entity with a NERC type, we could use the "category" or "type" information of the entity in the knowledge base. We don't have enough training data to learn all the possible combinations of NERC types and entities, but we have enough data to learn sufficient NERC type and entity category combinations.

The category information of entities is extracted from the existing knowledge bases introduced before, and represent a form of external background knowledge that makes possible detecting incompatible NERC and EL annotations. Please notice that, in the knowledge bases, an entity does not just belong to one category but many categories organized in a hierarchy structure. To simplify the problem, we account for the existence of a hierarchy by mapping an entity of some child category to all the parent categories in the hierarchy (i.e., we materialize typing information along the subclass relation). After this kind of inference, we may discard hierarchy relations and just deal with the sets of categories entities are assigned for. Also, some categories are rarely seen and too specific, which will lead to a category set that has very few, or even only one, entities. Considering this situation, we rule out such rare categories from all category sets and take into account only more general ones. After this, we have less categories to consider, less possible category sets to count, and more general category sets.

Every time a NERC type appears with an entity belonging to a certain set of categories, the "compatibility" between this NERC type and this set of categories increases.

In the meantime, we also need to consider that, the confidence score of the candidates themselves matters as well. For example, two Entity Linking candidates are both compatible with a certain NERC candidate, but one Entity Linking candidate has a high confidence score itself while the other one's confidence score is very low. In this situation, obviously we consider the first candidates as the more likely one.

Therefore, we propose a method to find the best combination of NERC and Entity Linking candidates under a same text mention. Provided that there

are several NERC and Entity Linking candidates for a mention, we estimate the joint probability of each possible pair of the NERC type and linked entity, which is the combination of the two candidates' confidence score and their compatibility score.

All the steps of processing above could be summarized as follows:

- Training stage
  - Go through all the Entity Linking results in dataset to count the frequency of entity categories. Categories with low frequency should be eliminated
  - For each NERC type/Entity Linking pair in dataset, record the concurrence of this NERC type and the entity's corresponding category set
  - Based on the data collected above and possibly other statistics computable from the knowledge base, fit the parameters of a probabilistic model able to estimate the joint probability of a NERC type and a linked entity (via its category set)
- Testing stage
  - Take every possible NERC type/Entity Linking candidate pair under a text mention, calculating their joint probability using the trained probabilistic model
  - The pair with highest joint probability is taken as final result and saved back in NAF file

3.2 The Probabilistic Model

The last section already discussed the general idea of the probabilistic model. In this section, we'll introduce in detail the mathematical derivation process of the model.

Let's start with a more general model, with n various kinds of linguistic annotations. For better demonstration, we employ a probability graphical model [10] to show the dependencies between variables.

The probability graph of this model is shown in Figure 6.

Given a mention m (a mention stands for a snippet denoting an entity and its textual context), there is the set of categories (classes) C and a set of annotations $\{a_1, a_2, \cdots, a_n\}$.

From the probability graph, it can be seen that in our modeling, category c depends on mention m, and annotations $a_1, a_2, \cdots, a_n$ all depend on both mention m and category C. In other words, if a certain mention is given, we have more knowledge about what categories it should belong to. When we already know both the mention and the categories, more knowledge about annotations are known. Of course, only a mention or only the categories can

tell a lot about the annotations, but the mention and categories enhance each other.

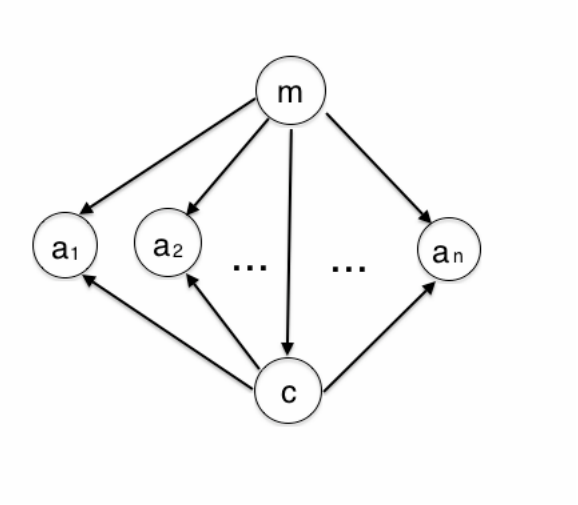


Figure 6: Probability graph of the general model

Let's take the mention *Liverpool* as example, as we did before. Knowing the mention, we can deduce that the possible categories should include something like *Cities_in_England* or *Football_clubs_in_England*, or even *Billing_authorities_in_England* (corresponding to the entity *Liverpool_City_Council*). Even more, we know that the NERC type of this mention is more likely to be *LOCATION* or *ORGANIZATION* and less likely to be *PERSON*. Moreover, if we know that the categories include *Cities_in_England*, we would be more sure that the NERC type should be *LOCATION*.

In the next step, we need to factorize the probability according to the graph.

The final result we need is the joint probability of all the variables in the graph, $P(m, C, a_1, a_2, \cdots, a_n)$, which could be easily written as:

$$P(m, C, a_1, a_2, \cdots, a_n) = P(m)P(C|m)P(a_1, \cdots, a_n|C, m) \tag{6}$$

According to the Bayes net assumption, each variable is conditionally independent of its non-descendants, given its parents. This is more formally defined as d-separation. In our graphical model, mention m and categories C are parents of all the annotations $a_1, \cdots, a_n$, and the annotations are not descendants of each other. Thus, annotations $a_1, \cdots, a_n$ are independent given mention m and categories C, written as:

$$P(a_1, a_2 \cdots, a_n|C, m) = \prod_{i=1}^{n} P(a_i|C, m) \tag{7}$$

Combining equation (6) and (7), we get:

$$P(m, C, a_1, a_2, \cdots, a_n) = P(m)P(C|m)\prod_{i=1}^{n} P(a_i|C, m) \tag{8}$$

Now, what needs to be done is to further decompose the following:

$$P(a_i|C, m) \tag{9}$$

With the formula of conditional probability:

$$P(A|B) = \frac{P(A, B)}{P(B)} \tag{10}$$

(9) can be further decomposed as:

$$P(a_i|C, m) = \frac{P(a_i, C, m)}{P(C, m)} \tag{11}$$

Here, we have to make another independency assumption that, given a specific annotation a_i, mention m and categories C are independent. That's because, when an annotation is given, the categories are almost settled and the mention can't give much more information to decide it.

So with this independency assumption and Bayes' Theorem, which could be written as:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{12}$$

(11) could continue to decompose:

$$\begin{aligned} P(a_i|C, m) &= \frac{P(a_i)P(C|a_i)P(m|a_i)}{P(m)P(C|m)} \\ &= \frac{P(m)P(a_i|m)P(C|a_i)}{P(m)P(C|m)} \\ &= \frac{P(a_i|m)P(C|a_i)}{P(C|m)} \end{aligned} \tag{13}$$

Until here, $P(C|m)$ is the only part needed to be factorized more, as $P(a_i|m)$ results from the NLP tools and $P(C|a_i)$ can be counted from datasets. To get $P(C|m)$, we can sum over all the annotations $a_1, \cdots, a_n$ and all the possible i-th annotations a_i in $P(a_i, C, m)$. Suppose that there are q_i possible candidates

for annotation a_i, written as $\{a_{i_1}, a_{i_2}, \cdots, a_{i_q}\}$, by summing up we get [1]:

$$\begin{aligned} P(C|m) &= \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{q_i} P(a_{i_j}, C|m) \\ &= \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{q_i} \frac{P(a_{i_j}, C, m)}{P(m)} \\ &= \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{q_i} \frac{P(a_{i_j})P(C|a_{i_j})P(m|a_{i_j})}{P(m)} \\ &= \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{q_i} \frac{P(m)P(a_{i_j}|m)P(C|a_{i_j})}{P(m)} \\ &= \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{q_i} P(a_{i_j}|m)P(C|a_{i_j}) \end{aligned} \tag{14}$$

We can now put the result of (13) to the very first equation (7) and get:

$$\begin{aligned} P(m, C, a_1, a_2, \cdots, a_n) &= P(m)P(C|m)\prod_{i=1}^{n}\frac{P(C|a_i)P(a_i|m)}{P(C|m)} \\ &= P(m)P(C|m)\frac{1}{P(C|m)^n}\prod_{i=1}^{n}P(C|a_i)P(a_i|m) \\ &= P(m)\Big(\frac{1}{P(C|m)}\Big)^{n-1}\prod_{i=1}^{n}P(C|a_i)P(a_i|m) \end{aligned} \tag{15}$$

After getting to know with the more general probability model, we can specify it according to our task. Mention m and categories C are kept, while the annotations $a_1, a_2, \cdots, a_n$ are replaced by NERC type t and Entity Linking e, as shown in Figure 7.

By replacing $a_1, a_2, \cdots, a_n$ with t and e in (15), we have the following equation:

$$P(m, C, e, t) = P(m)\frac{1}{P(C|m)}P(C|e)P(e|m)P(C|t)P(t|m) \tag{16}$$

where

$$P(C|m) = \frac{1}{2}\Big(\sum_{j=1}^{q_e}P(e_j|m)P(C|e_j) + \sum_{j=1}^{q_t}P(t_j|m)P(C|t_j)\Big) \tag{17}$$

and

$$P(C|e) = \begin{cases} 1 & \text{if } e \text{ has category set } c \\ 0 & \text{otherwise} \end{cases} \tag{18}$$

[1] This part of derivation could also be replaced by choose a specific a_i and do the estimate based on that a_i. Averaging over all the annotations aims at producing a more accurate estimate of $P(C|m)$

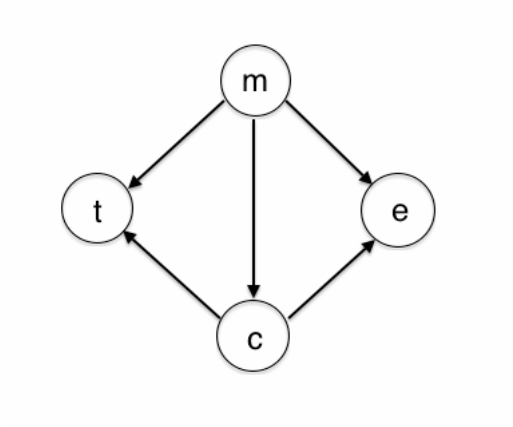


Figure 7: Probability graphical model with NERC type and Entity Linking

In the situations that $P(C|e) = 0$, i.e., C is not the category set of entity e, it's obvious from (16) that the joint probability is 0, so we only consider situations that c is the category set of e [2].

Our job is to find the best NERC type/Entity Linking combination that maximizes the joint probability of $P(m, C, e, t)$.

3.3 Construction of the Model from the Dataset

First, we need to explain a little more about the pre-processing of entity categories.

For the time being, we adopt the category system of YAGO, which is more numerous and fine-grained than the one of DBpedia.

It's already explained before that the categories of an entity, although organized in a hierarchy structure, are treated as a set in the model.

Also, the categories appearing less than specific times in the dataset are ignored. A threshold K is set for this appearing times and only categories appearing more than K times in the dataset are taken into consideration.

The specific steps for getting category sets are as follows:

1. From the dataset, constructing a text file "mention-categories", where each line contains a mention and all its Entity Linking's corresponding categories are extracted from YAGO

2. Based on the "mention-categories" file, counting the appearing times of each category and saving the result in a "class-count" file

[2]This is a simplification that does not account for the fact that categories in the KB may be incomplete (open world assumption), and thus it may lead to a non-zero probability for entity e having a category set larger than the known set of categories assigned to e.

3. From YAGO database and the annotated dataset, building a file "mention-ner-el-categories", where each line has a mention, its NERC type, its Entity Linking and the categories of its Entity Linking

4. Based on "mention-ner-el-categories" file, depending on different K values, with information in "class-count" file, writing different versions of files with same structure as "mention-ner-el-categories" file but deleting categories appearing less than K times in dataset

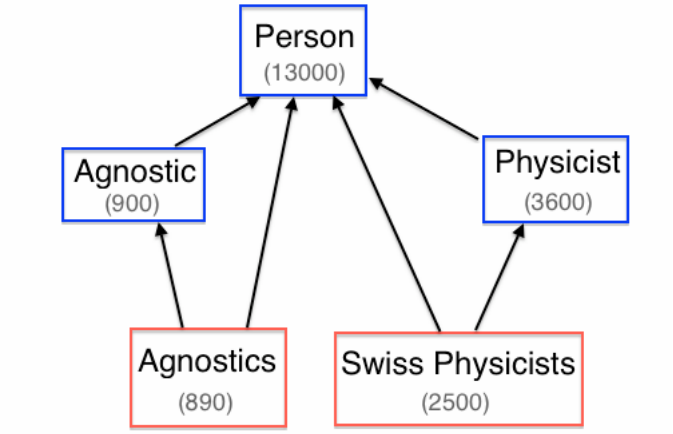


Figure 8: Part of category hierarchy of entity *Albert Einstein*. Categories with blue frame are from WordNet and the ones with red frame are from Wikipedia. Arrows point from subclasses to higher classes. Numbers below category names are appearing times of that category in dataset

Figure 8 shows an example of category hierarchy in YAGO. The original category set for this entity *Albert Einstein* is {*Swiss Physicists, Agnostics, Physicist, Agnostic, Person*}. Please note that, the first two categories come from Wikipedia, the latter three come from Wordnet, and that's why there exists two "Agnostics" in the set. Suppose that K is set to 1000 in the model, meaning that categories appearing less than 1000 times in dataset should be eliminated. As categories *Agnostic* and *Agnostics* appeared only 900 and 890 times in dataset, they're ignored for later calculation. Therefore, the category set for this entity should be {*Swiss Physicists, Physicist, Person*}.

After getting the category set with frequent categories for Entity Linking candidates, the next step is to get all the probabilities needed in equation (16) and (17). Details are explained one by one as listed below:

- $P(m)$: It's the same for all NERC type/Entity Linking combinations, so it could be ignored

- $P(c|e)$: As mentioned before, this value can only be 0 or 1, and we only consider the situation that it's 1, where the catetory set is the one of the entity, as acquired directly from YAGO using specific queries and cut down by looking at file "class-count"
- $P(e|m)$: Given by the Entity Linking tool, DBpedia Spotlight, and presented as confidence score with the Entity Linking candidate
- $P(t|m)$: Given by the Stanford NERC tool, presented as confidence score with the NERC candidate
- $P(c|t)$: Counted from "mention-ner-el-categories" file. Going through all the mentions with a specific NERC type and counting how many times it appears with a specific category

3.4 Implementation

The model is Java-based and uses Maven for project management and comprehension.

In this chapter, we'll introduce the important classes we created and which task they achieve. The following list covers the names of the classes and the description:

- Class_Sort: Preparing needed files
  - Take "mention-categories" file to count the appearing time of each category, sort the result according to appearing time and save it to "class-count" file
  - According to different K values, create different "mention-ner-el-categories" files, such as "mention-ner-el-categories_K1000" with categories appearing more than 1000 times in dataset
- Get_Probabilities: Realizing the probabilistic model
  - Take three HashMap as input: one contains Entity Linking candidates with confidence scores, another contains NERC type candidates with confidence scores, the last one contains Entity Linking candidates with their category sets
  - Go through the "mention-ner-el-categories" file and calculate the probability $P(c|t)$ for each NERC type/category set pair
  - Compute the joint probability as equation (16) and (17) for every NERC type/Entity Linking pair
  - Return the NERC type/Entity Linking pair with the highest joint probability
- Parse_Annotation: Dealing with NAF files

  - Parse the NAF file produced by PIKES and find all the mentions with both NERC type and Entity Linking annotations
  - Send a query to YAGO and get the category information of all the Entity Linking candidates
  - According to K value, delete categories that have less than K counts in "class-count" file
  - Send the needed information to class *Get_Probabilities* and get the returned best NERC type/Entity Linking pair for all mentions
  - Save the best NERC and Entity Linking candidates for all mentions back to NAF file, and the confidence score for the best candidates are set to 2.0

- Create_Evaluation: Creating files with correct format required by evaluation scripts
  - Parse NAF files in order (there are 1393 NAF files with file name from "0001" to "1393", they need to be read in the same order of their file names) after modification of last step, go through all entities and find the NERC type or Entity Linking candidate with highest confidence score
  - For NERC type evaluation, create similar files as Figure 1. Each line contains token, POS tag, phrase structure and NERC type. If a token does not correspond to any NERC type, the last column would be "O".
  - For Entity Linking evaluation, create files similar to the one shown in Figure 2. Each line contains token, mention span, mention text and entity identifier. If a token does not correspond to any Entity Linking, only the token would be written in the line.

K	correct phrase	accuracy	precision	recall	F_1
Baseline	32088	98.32%	94.76%	92.11%	93.42
1000	31477	98.01%	92.99%	90.36%	91.65
2000	31662	98.11%	93.53%	90.89%	92.19
3000	31933	98.24%	94.32%	91.67%	92.97
5000	31941	98.25%	94.34%	91.69%	93.00
7000	31945	98.25%	94.35%	91.70%	93.01
10000	31980	98.27%	94.45%	91.80%	93.11

Table 1: Overall result for baseline and model with different K values

4 Evaluation

In our probabilistic model, there's one parameter to be set, the K value, which is the threshold for category appearing time. Only categories which appeared more than K times in the dateset are considered into calculation.

By counting appearing times of all the categories, we recorded the result in "class-count" file. In this file, there are more than 20038 categories, and most of them appeared only a few times. When K is set to 1000, there are 79 categories left; when K equals 10000, there are only 10 left.

In order to reduce over-fitting and amount of calculation, we set a few different values from 1000 to 10000 for K, and they are: 1000, 2000, 3000, 5000, 7000, 10000. In all the tables or figures of this chapter, the "baseline" result refers to the evaluations result without the probabilistic model, i.e., the original result produced by PIKES.

4.1 NERC Result

As introduced before, the evaluation result produced by the script include information like total phrase numbers, correct phrase numbers, accuracy, precision, recall and F_1 score for overall and respective types.

Table 1 shows the NERC result with different K values and also the baseline, while Table 2 shows the result of different K values on four different NERC types separately.

For the purpose of making statistical data more intuitive, two charts are provided. The one shown in Figure 9 represents the overall performance of baseline and the model. For Figure 10, instead of showing all different kinds of scores, the score of F_1, which is computed by both the precision and recall score, is chosen for representation.

From both Table 1 and Figure 9, it could be seen that, all precision, recall and F_1 have almost exactly the same trend. At baseline, the system performs quite well. When $K = 1000$, the performance is the worst, the scores decreased about 2% compared to the baseline. After that, as K increases, the perfor-

NERC	K	precision	recall	F_1	correct phrase
LOC	baseline	97.27%	94.56%	95.89	10314
	1000	91.07%	90.36%	94.52	10558
	2000	95.20%	90.89%	94.86	10535
	3000	96.78%	91.67%	95.71	10376
	5000	96.80%	91.69%	95.69	10369
	7000	96.79%	91.70%	95.70	10372
	10000	96.70%	91.80%	95.73	10397
MISC	baseline	92.80%	88.18%	90.43	4725
	1000	91.07%	88.18%	89.60	4810
	2000	91.71%	88.30%	89.97	4788
	3000	92.00%	88.30%	90.11	4773
	5000	92.16%	88.38%	90.23	4769
	7000	92.14%	88.44%	90.25	4773
	10000	92.63%	88.40%	90.46	4746
ORG	baseline	94.37%	91.99%	93.16	9025
	1000	91.80%	91.02%	91.41	9181
	2000	92.68%	91.36%	92.02	9127
	3000	93.25%	92.10%	92.68	9145
	5000	93.25%	92.20%	92.72	9155
	7000	93.27%	92.19%	92.73	9152
	10000	93.53%	92.22%	92.87	9130
PER	baseline	93.43%	91.59%	92.50	9798
	1000	93.15%	86.65%	89.78	9297
	2000	93.41%	87.88%	90.56	9403
	3000	93.82%	89.77%	91.75	9563
	5000	93.80%	89.77%	91.74	9565
	7000	93.84%	89.77%	91.76	9561
	10000	93.79%	89.96%	91.84	9586

Table 2: Evaluation result for respective NERC types

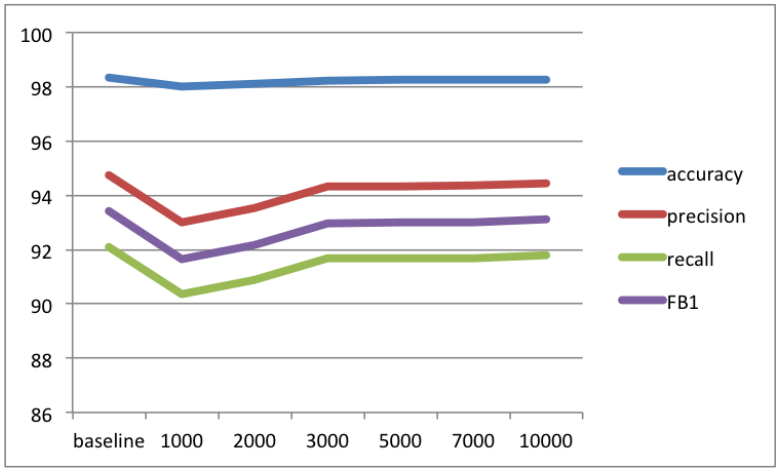


Figure 9: Accuracy, precision, recall and F_1 result for NERC type. The score of the first three are percentages(%), while F_1 score is a number.

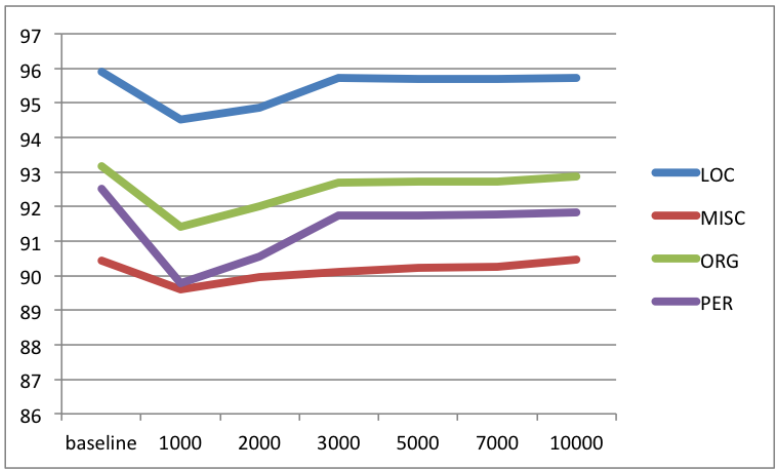


Figure 10: F_1 score of the four different NERC types.

mance improves again. After K arrives at the value of 3000, there's still slight improvement, but in general the performance stays almost the same as baseline.

Table 2 and Figure 10, however, shows another aspect of the result.

Comparing among the four different NERC types, it's very obvious that type *LOCATION* has the best performance, not only for F_1 but also for other kinds of measures including precision and recall, except for the recall score from the model, which is a little lower than the one of *ORGANIZATION*. Also, as seen from F_1 score, the model has the second best performance on *ORGANIZATION*, then follows *PERSON*, and type *MISC* has the lowest score. It's interesting to

K	TP	FP	FN	precision	recall	F-Score
Baseline	16389	31946	11428	0.339	0.589	0.430
1000	16382	31953	11435	0.339	0.589	0.430
2000	16586	31749	11231	0.343	0.596	0.436
3000	16987	31348	10830	0.351	0.611	0.446
5000	16932	31403	10885	0.350	0.609	0.445
7000	16910	31425	10907	0.350	0.608	0.444
10000	16805	31530	11012	0.348	0.604	0.441

Table 3: *Strong link match* result for baseline and model with different K values

notice that, this also corresponds to the "correct phrase" number: the type that appears more has better performance.

What's more, comparing Figure 9 and Figure 10, we can find that the line charts still have the same trend. When $K = 1000$ and $K = 2000$, the model have worse performance, and at baseline and when $K > 3000$, the performance is very similar.

4.2 Entity Linking Result

In chapter 2.3.2, it's mentioned that the evaluation script for Entity Linking produces four kinds of match evaluation, namely *weak mention match*, *strong mention match*, *strong link match* and *entity match*. It's also explained that the first two are about mention detection, and because the probabilistic model changes only the linked entity but not the mention span, they're always the same with or without the model, so it's not of much use paying attention to these two match evaluations. Therefore, we only focus on *strong link match* and *entity match*. Between these two, our focus is more on *strong link match*, because it's more precise.

Table 3 and Table 4 show the evaluation result of *strong link match* and *entity match*. These two tables have the same organization: including TP (true positive), FP (false positive), FN (false negative), precision, recall and F-score.

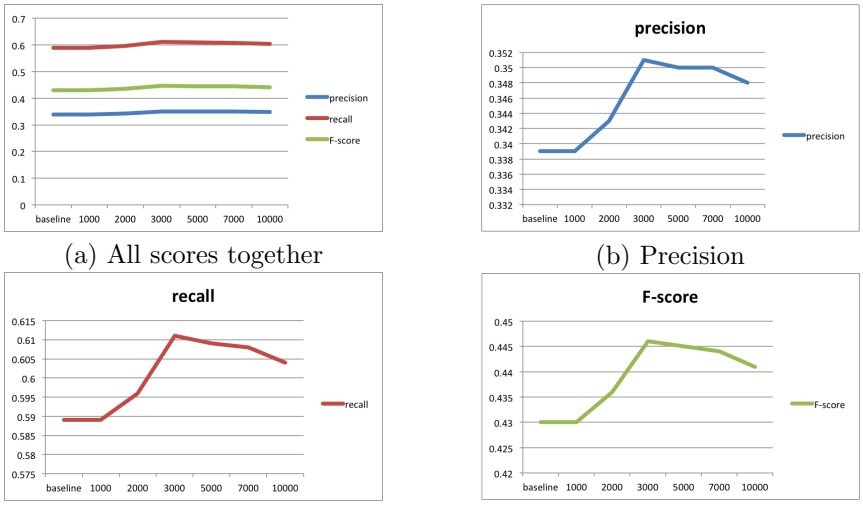
Also, to make the data more intuitive to see and understand, we made a line chart depending on *strong link match* shown in Figure 11.

The first image of the figure is three evaluation scores together. For the fact that the difference between evaluation measures are much greater than the difference between different parameter within the same evaluation measure, the following three images are made separately to make everything more clear.

From the figure, we can see that the result of precision, recall and F-score all have the same trend. From the beginning (baseline), the performance improves fast as K value increases. At the point where $K = 3000$, the model has the best result. Since then, the score continues to decrease slowly. Depending on this information, it can be said that, if judged by *strong link match* performance,

K	TP	FP	FN	precision	recall	F-Score
Baseline	3769	6930	1826	0.352	0.674	0.463
1000	3768	7055	1827	0.348	0.673	0.459
2000	3772	7014	1823	0.350	0.674	0.461
3000	3768	6974	1827	0.351	0.673	0.461
5000	3774	6967	1821	0.351	0.675	0.462
7000	3772	6970	1823	0.351	0.674	0.462
10000	3743	6958	1852	0.350	0.669	0.459

Table 4: *Entity match* result for baseline and model with different K values



(a) All scores together

(b) Precision

(c) Recall

(d) F-score

Figure 11: Precision, recall and F-score for *strong link match*

K	Right to Wrong	Wrong to Right
1000	1176	255
2000	892	257
3000	477	253
5000	444	229
7000	436	225
10000	371	218

Table 5: Number of NERC changed from correct one to wrong one or from wrong to correct after applying the model

the best K value for the probabilistic model is 3000. At this point, presision, recall and F-score all increase around 0.015.

As for *entity match*, by roughly looking at Table 4, one can easily conclude that, with or without the model, or no matter how K value changes, the result is not affected a lot, and the best K value may be around 5000.

4.3 Analysis

The main results of the model is already shown in the last two chapters. Here we'll do some further analysis based on that.

The model did well on improving the Entity Linking result, but does not seem to help a lot on NERC. The best the model could do is to get the same performance as baseline.

The reason behind this phenomenon is not hard to find if we compare the NERC and Entity Linking results. For NERC, all the scores are above 90, even in the worst case where $K = 1000$, however, for Entity Linking, the scores are much lower. From common sense, it's easy to understand that it's more difficult to improve a system that is already performing very well than a less perfect system. Moreover, in our model, when calculating the joint probability of all NERC type/Entity Linking pairs, Entity Linking could "benefit" more from NERC, which is more accurate; reversely, NERC could not "benefit" too much from the less accurate Entity Linking.

Table 5 and 6 are showing the "change" information for NERC and Entity Linking after applying the model. They demonstrate the number of mentions whose NERC type and Entity Linking are changed from a correct one to a wrong one or from a wrong one to a correct one after the model.

It's not a surprise to get the result in Table 5 and 6. Many Entity Linking are changed correctly, more than the number of ones changed from correct to wrong, while NERC has a totally different situation.

To further explore this, we did another experiment. Different from the model before, we tried to make the NERC type fixed and change only the Entity Linking. In other words, in this experiment, under a mention, we only consider the NERC type candidate that is originally the best, with the highest confidence score. We only run the experiment with $K = 3000$, which is the best parameter

K	Right to Wrong	Wrong to Right
1000	677	659
2000	409	604
3000	168	787
5000	139	701
7000	114	654
10000	296	730

Table 6: Number of Entity Linking changed from correct one to wrong one or from wrong to correct after applying the model

NERC type	TP	FP	FN	precision	recall	F-Score
Top candidats only	16987	31348	10830	0.351	0.611	0.446
All candidates	16994	31341	10823	0.352	0.611	0.446

Table 7: Result of Entity Linking when $K = 3000$. The first line is the result produced from the model as before, and the second line is the result from the model in which NERC types are fixed

value according to former result, instead of running it on all K values. The result of this experiment is shown in Table 7.

From Table 7, we can see that in a model where NERC type is fixed to the candidate with highest confidence score, the result of Entity Linking does improve, although the improvement is negligibly small.

5 Discussion and Conclusion

5.1 Conclusion

In this work, we explored a probabilistic model to extract the most coherent NERC/Entity Linking pair under a same text mention.

The model mainly took advantage of two resources: category information of linked entities from existing knowledge bases and CoNLL-AIDA 2013 dataset. The former is to generalize the concurrence. Instead of counting directly the times a NERC and Entity Linking appear together, we consider the concurrence of a NERC and a category set. The latter serves as training set and gold-standard.

In order to avoid over-fitting, we eliminated the relatively rare categories and kept only the frequent ones. We set a parameter K, categories appearing less than K times in dataset are considered "not frequent" and are ignored. We ran several experiments with different K value and compared the results. It turned out that, when K is smaller, performance of the system is worse, and as K increases, the system performs better. After a certain point, even if K continues to grow, the performance does not change much.

We also found that Entity Linking benefited much more and NERC did from the model. This is probably due to the huge difference of accuracy between NERC and Entity Linking at the beginning. The system reached a high score for NERC, so there didn't exist much room for improvement, while the original performance of Entity Linking was far from perfect, which made it more possible to improve.

5.2 Future Work

Based on this project, there are some further works possible to be done in the future:

- Different tools for NERC and Entity Linking can be applied before running the model. PIKES takes a pipeline structure for producing different annotations, so a pipeline for a certain task can be easily replaced by another. The probabilistic model needs only the output of a pipeline, thus, it would still work with other annotation tools

- In this project we adopted the category system of YAGO, but it's also possible to adopt the one of DBpedia. We can do this experiment and see if the result changes.

- More kinds of annotations can be included in the model. Here, we only include NERC and Entity Linking, but we have already derived a model with n different kinds of annotations

References

[1] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. *The semantic web*, pages 722–735, 2007.

[2] Linguistic Data Consortium et al. Message understanding conference (muc) 7. *LDC2001T02. FTP FILE. Philadelphia: Linguistic Data Consortium*, 2001.

[3] Francesco Corcoglioniti, Marco Rospocher, and Alessio Palmero Aprosio. A 2-phase frame-based knowledge extraction framework. In *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, pages 354–361. ACM, 2016.

[4] Marco Cornolti, Paolo Ferragina, and Massimiliano Ciaramita. A framework for benchmarking entity-annotation systems. In *Proceedings of the 22nd international conference on World Wide Web*, pages 249–260. ACM, 2013.

[5] Joachim Daiber, Max Jakob, Chris Hokamp, and Pablo N Mendes. Improving efficiency and accuracy in multilingual entity extraction. In *Proceedings of the 9th International Conference on Semantic Systems*, pages 121–124. ACM, 2013.

[6] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 363–370. Association for Computational Linguistics, 2005.

[7] Stephen Guo, Ming-Wei Chang, and Emre Kiciman. To link or not to link? a study on end-to-end tweet entity linking. In *HLT-NAACL*, pages 1020–1030, 2013.

[8] Xianpei Han and Le Sun. A generative entity-mention model for linking entities with knowledge base. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 945–954. Association for Computational Linguistics, 2011.

[9] Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. Robust disambiguation of named entities in text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 782–792. Association for Computational Linguistics, 2011.

[10] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*. The MIT Press, 2009.

[11] David D Lewis, Yiming Yang, Tony G Rose, and Fan Li. Rcv1: A new benchmark collection for text categorization research. *Journal of machine learning research*, 5(Apr):361–397, 2004.

[12] Gang Luo, Xiaojiang Huang, Chin-Yew Lin, and Zaiqing Nie. Joint named entity recognition and disambiguation. In *Proc. EMNLP*, pages 879–880, 2015.

[13] George Miller and Christiane Fellbaum. Wordnet: An electronic lexical database, 1998.

[14] Avirup Sil and Alexander Yates. Re-ranking for joint named-entity recognition and linking. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 2369–2374. ACM, 2013.

[15] Rosa Stern, Benoît Sagot, and Frédéric Béchet. A joint named entity recognition and entity linking system. In *Proceedings of the Workshop on Innovative Hybrid Approaches to the Processing of Textual Data*, pages 52–60. Association for Computational Linguistics, 2012.

[16] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706. ACM, 2007.

[17] Erik F Tjong Kim Sang and Fien De Meulder. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 142–147. Association for Computational Linguistics, 2003.