NTUNLP Approaches to Recognizing and Disambiguating Entities in Long and Short Text in the 2014 ERD Challenge

Yen-Pin Chiu, Yong-Siang Shih, Yang-Yin Lee, Chih-Chieh Shao,
Ming-Lun Cai, Sheng-Lun Wei, Hsin-Hsi Chen

Natural Language Processing Laboratory

Department of Computer Science and Information Engineering

National Taiwan University, Taipei, Taiwan

+886-2-33664888ext301

{r02944035, r02922036, d02922015, r02944040, b00902096, b99501062, hhchen}@ntu.edu.tw

ABSTRACT

This paper presents the NTUNLP systems in the long track and the short track of the Entity Recognition and Disambiguation Challenge 2014. We first create a dictionary that contains the possible surface forms of Freebase Ids, then scan the given text from left to right with the longest match strategy to detect the mentions, and eliminate the unwanted surface forms based on a stop word list. Methods to link to the most relevant entities and select the best candidate are proposed for these two tracks, respectively. The outside resources such as DBpedia Spotlight and TAGME are integrated to our basic NTUNLP systems. Various experimental setups are presented and discussed with the development set. In the formal run, one NTUNLP system wins the first prize in the short track and another NTUNLP system gets the fourth place in the long track.

Categories and Subject Descriptors

H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing – *Linguistic processing*

General Terms

Algorithms, Experimentation.

Keywords

NTUNLP, Entity Recognition, Entity Disambiguation, TAGME, DBpedia Spotlight.

1. INTRODUCTION

The demand to recognize and disambiguate entities in a given search query has been raised for years. This effort can bring lots of benefits in many researches in natural language processing and information retrieval, such as effective document searching, machine translation, knowledge graph construction and question answering. The key issues are how to segment queries into correct mentions (extraction phase), and how to choose appropriate named entities for a specific mention (disambiguation phase). The goal of the Entity Recognition and Disambiguation Challenge (ERD) [1] under the SIGIR 2014 conference is to find mentions in a given text and link the mentions to the corresponding entities in a given entity dataset. The challenge is composed of two tracks: (1) the short text and (2) the long text. In the short track, search queries are given as short text, while long documents are given in the long track. Besides the length difference, another different aspect between these two tracks is that there may be multiple interpretations of a given query in the short track, while there is only one interpretation per mention in a document in the long track. In the ERD 2014, our NTUNLP team is the winner of the first prize in the short track and gets the fourth place in the long track. That demonstrates the effectiveness and robustness of our systems in the short and long tracks.

In the long track, we extend the original entity dictionary provided by the organizers to cover all possible surface forms of each entity's Freebase Ids. Furthermore, we use the 2014-03-04 Wikipedia data dump to extract redirect links of the entities. The final dictionary in terms of the possible surface forms of Freebase Ids and the extracted redirect links contains 5,846,172 entries. Then we read the long text from left to right and use the longest common subsequence matching algorithm to detect the possible mentions. We also construct a stop word list to eliminate unwanted surface forms. After stop word removal, we apply the Freebase search API on every candidate of a mention to get the most relevant entity. Finally, we integrate the basic NTUNLP system with DBpedia Spotlight [2] and TAGME [3], and achieve the expected F1 score 0.7137, precision 0.7571, and recall 0.6750 in the formal run.

In the short track, our NTUNLP system segments a query text into mentions and context words by string matching between a controlled vocabulary and the substrings of the query. And then we filter out some inappropriate mentions by a stop word list based on the commonness of the words. We further integrate the basic system with TAGME and use the Wikipedia article title to further improve the system performance. In the development set provided by the organizers, we get the expected F1 score 0.6621. In the formal run, we achieve the expected F1 score 0.6797.

The remainder of this paper is organized as follows. Section 2 gives a survey on the related work. Section 3 describes the methods we used in the long track. Section 4 reports the experimental results of the long track. The methods of the short track are presented in Section 5. Section 6 reports the experimental result of the short track. Section 7 concludes the remarks and describes the future work.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.
ERD'14, July 06 - 11 2014, Gold Coast, QLD, Australia
Copyright 2014 ACM 978-1-4503-3023-7/14/07…$15.00.
http://dx.doi.org/10.1145/2633211.2634363

2. RELATED WORKS

In the recent years, several works have been proposed to identify the named entities in an input text via a structured and linked knowledge database such as Wikipedia and Freebase. Furthermore, they also annotate those named entities with URLs that link to the Wikipedia page.

The system Wikify [4] is one of the early works on the entity linking task. It automatically labels a link to Wikipedia page of the input text. In Wikify, Mihalcea and Csomai define the *text wikification* problem as a task that automatically extracts the important words and phrases from a given text, and then maps each keyword to an appropriate Wikipedia page. The most important two subtasks are: (1) to extract keywords from the text, and (2) to disambiguate the candidate links. In the keyword extraction phase, they construct a keyword vocabulary that contains only the words in the Wikipedia article titles, and then uses this controlled vocabulary to extract keyphrases. In the link disambiguation phase, they use a knowledge-based method, a feature-based method and their combination to compute the similarity between the candidate Wikipedia articles and the input context.

In 2008, Milne and Witten [5] propose a machine learning approach that utilizes the Wikipedia knowledge base as an information source. They extract two features from the Wikipedia article corpus. The *commonness* feature describes the number of in-links of a Wikipedia article page and the *relatedness* feature describes the in-link overlap level between candidate Wikipedia pages. Their work gains a magnificent improvement and achieves an F-Measure of 74.8%.

In 2010, TAGME system [3] proposed by Ferragina and Scaiella efficiently adds a plain-text with pertinent hyperlinks to the Wikipedia pages. The basic idea follows the work of Milne and Witten, but they introduce the *relatedness* feature of the ambiguous keywords into training instead of only considering the *relatedness* feature of the unambiguous keywords. TAGME system focuses on the short text and outperforms the previous works. Moreover, their work retains its competitive advantage on the long text too. TAGME system yields an F-measure of 77.9%.

DBpedia spotlight [2] uses the extended set of labels in the DBpedia lexicalization dataset to create a lexicon for spotting, and uses the Wikipedia dataset to disambiguate the named entities by Vector Space Model (VSM) [6]. In VSM, each DBpedia context is regarded as a point in the multidimensional space of terms. Unlike the traditional TF-IDF weighting scheme, DBpedia spotlight introduces the TF-ICF (Inverse Candidate Frequency) weighting and demonstrates TF-ICF weighting outperforms TF-IDF weighting.

Besides the above systems, some researches focus on finding the candidate entities in the context. Hakimov [7] uses the greatest common subsequence to match entity candidates and uses each candidate's POS tags to filter out non-suitable entities for named entity disambiguation. LingPipe Exact Dictionary-Based Chunker[1] uses the Aho-Corasick string matching algorithm [8] with the longest case-insensitive match. The NERSO system [9] introduces the disambiguation method based on page links between all spotted named entities, and assigns string similarity score between the spotted entities and the named entity to be disambiguated. Fader et al. [10] choose the most popular candidate whose contextual evidence is higher than a threshold.

As for the knowledge base, Wikipedia is used by many research teams, such as Bunescu and Pasca [11], Cucerzan [12], Bizer et al. [13], Mihalcea and Csomai (Wikify!) [4], Witten and Milne (M&W) [5] and, Mendes (DBpedia spotlight) [2], and almost all of the works use VSM to measure the similarity between difference contexts.

3. LONG TRACK

3.1 Constructing a Dictionary

To detect entities in the long text, the possible surface forms must be known. To this end, we create a dictionary that contains the possible surface forms of Freebase Ids. Some example entries are shown as follows.

```
/m/0_007 Cass Township
/m/0_007 Cass Township, Huntingdon County, PA
/m/0_007 Cass Township, Huntingdon County, Pennsylvania
/m/0_00l  Cassville
/m/0_00l  Cassville, PA
```

The same Freebase Id may be mapped to multiple surface forms, and it is also possible to link different Freebase Ids to the same surface form. For example:

```
/m/02h39h "New York" (New York, Tyne and Wear)
/m/04c77h "New York" (New York, Lincolnshire)
/m/04jlxvb "New York" (a film)
/m/059rby "New York" (New York State)
...
```

The original entity.tsv[2] provided by the organizers only contains one surface form for each entity, so we extract alias of entities using Freebase data dump from 2014-03-23 [14] by the "/common/topic/alias" relation. Furthermore, we use the 2014-03-04 data dump from Wikipedia to extract redirect links and anchor texts of the entities. However, our preliminary experiments show that precision would be significantly reduced when anchor texts are included, so only the redirect links are included and anchor texts are not used in the end. The final dictionary we constructed contains 5,846,172 entries.

3.2 Detecting Surface Forms

Next we scan the long text from left to right and use the longest match with the dictionary constructed in Section 3.1 to detect the possible surface forms. The longest match strategy is adopted because the time complexity is too high to explore all the possible combinations of n-grams. To increase the precision, we only consider those exact matches to the dictionary. That is, all symbols and letter cases must be matched exactly.

For example, if the dictionary contains the following entry:

```
/m/014kx2 King's X
```

then none of `King X`, `king's x`, `King` or `Kings X` would be proposed. Only `King's X` in the long text would be matched.

At this stage, a lot of non-entities words exist in the candidate list. Consider the following example. The long text begins with some words.

[1] http://alias-i.com/lingpipe/

[2] http://web-ngram.research.microsoft.com/erd2014/Docs/entity.tsv

VICTORIA CERAMICS JAPAN BOXING BOXER DOGS - Vintage Salt and Pepper S&P Shakers

We find the most interesting Vintage Salt.

Our system would detect the following surface forms, where the numbers are the offsets:

JAPAN	(18, 23)
Vintage	(44, 51)
Salt and Pepper	(52, 67)
S&P	(68, 71)
We	(80, 82)
find	(83, 87)
Vintage	(109, 116)
Salt	(117, 121)

Clearly, "We" and "find" are not entities. They are detected because there is a novel called "We" and there is also a command-line utility called "find" in our dictionary. We decide to remove this kind of entities that rarely exist in most documents. Therefore, a rule-based filtering is performed to eliminate the unwanted entries.

3.3 Eliminating Unwanted Surface Forms

Firstly, surface forms which contain only 1 character are discarded. This could eliminate some common words such as "I". Secondly, surface forms whose lengths are less than or equal to 4 and consist of only digits are discarded. This is because we find that many numbers such as years are wrongly detected as surface forms. Finally, we use a stop word list to further filter out some candidate surface forms. When matching with the stop word list, we treat the first letter case-insensitive, because the beginning of a sentence would be in uppercase. For example, our stop list contains the entry "we". Then, the following surface forms would be eliminated: "We" and "we". But "WE" would be preserved because it has an upper-case E, which is different from our stop list entry.

The stop word list is compiled from two sources: (1) a list of the top 5,000 words/lemmas from the 450 million word Corpus of Contemporary American English[3], and (2) MySQL 5.1 Full-Text Stopwords[4]. We manually selected 3,033 potential common words such as "won't", "wouldn't", "zero", and "baby" among these two lists to be our stop words.

3.4 Linking to the Most Relevant Entity

After the stop word removal, several surface forms still remain. For the previous example, we have:

JAPAN	(18, 23)
Vintage	(44, 51)
Salt and Pepper	(52, 67)
S&P	(68, 71)
Vintage	(109, 116)

We will link these surface forms to the correct entity. To this end, we use the Freebase search API with each surface text to get the most relevant entity. Given a query term, Freebase API would return a list of possibly matching entries, ranked by the relevancy score. For example, if we use "JAPAN" as the query term, the top three entries are shown as follows. The other additional information is not used.

Freebase Id	score
/m/03_3d	188.255219
/m/0gh6mkc	107.830437
/m/0c7fr	105.83

In this case, we select /m/03_3d, which is the most relevant one. We search the dictionary for a matching entry with this id and the original surface form. If there is an entry in the dictionary that has the same surface form and Freebase Id, then it is considered to be correct. For example, the following Freebase Id and surface form exist in our dictionary, so it would be accepted by our system.

/m/03_3d	JAPAN

Suppose the most relevant entity found by the Freebase Search API is shown as follows.

/m/0c7fr	105.83

Then we will try to find the following entry in our dictionary:

/m/0c7f	JAPAN

However, our dictionary only contain the following items:

/m/0c7fr	Billboard
/m/0c7fr	Internet Albums
/m/0c7fr	Japan Billboard Chart
/m/0c7fr	Japan
/m/0c7fr	Japanese Albums Chart
/m/0c7fr	Japan Hot 100
/m/0c7fr	Japan Hoy 100
/m/0c7fr	Jazz Albums
…	

So this entity would be rejected. That is, Freebase Id and surface form must both match so as to be accepted.

After considering the most relevant entity for each surface form, we would get two lists: one with those surface forms linked to their corresponding accepted entities; one with those surface form fail to link to an entity. For the previous example, we get:

JAPAN	(18, 23)	/m/03_3d
Vintage	(44, 51)	/m/08f97q
Vintage	(109, 116)	/m/08f97q

and

Salt and Pepper	(52, 67)
S&P	(68, 71)

3.5 Checking the Second Most Relevant Entity

Because the most relevant entity is likely to be the correct entity the surface form refers to, this process can eliminate incorrect and non-entity surface forms. However, sometimes the correct entity is actually the second most relevant one returned by Freebase Search API. It would be desirable if we can retain those surface forms too. Unfortunately, if we simply take the second most relevant entities for those surfaces forms not yet linked to an entity, and accept all the matching pairs in our dictionary just like we do for the most relevant entity, the precision would be decreased too much.

Therefore, we adopt a more conservative approach. We check the second most relevant entities for those unlinked surface forms

[3] http://www.wordfrequency.info/free.asp

[4] http://dev.mysql.com/doc/refman/5.1/en/fulltext-stopwords.html

using the same procedure as before, and only accept those Freebase Ids that already appear in our accepted Freebase Id set.

For example, in the previous example, the already accepted Freebase Id set is:

{/m/03_3d, /m/08f97q}

When we check the second most relevant entities for 'Salt and Pepper' and 'S&P', we accept them only when the Freebase Id is in the set {/m/03_3d, /m/08f97q}. In this case, the second most relevant entity for "Salt and Pepper" is /m/0bhxtp, and

/m/0bhxtp Salt and Pepper

actually exists in our dictionary. However, /m/0bhxtp is equal to neither "/m/03_3d" nor "/m/08f97q", so that it is still rejected.

The rationale behind this is that the same entity is likely to occur multiple times in a long text while the surface forms may change from one occurrence to another occurrence. We believe this procedure can help us catch those entities. Actually, this increases recall slightly while keeping the precision the same. Thus the strategy is shown to be effective in the development set.

3.6 Merging Results from DBpedia Spotlight and TAGME

To further increase the recall, we decide to include the results from the following two systems: DBpedia Spotlight [2] and TAGME [15]. Both systems can detect entities in a given text, i.e., they do the same thing as our system.

3.6.1 DBpedia spotlight

The DBpedia Spotlight would annotate the long text with URIs in DBpedia. For example, if the long text begins with:

Grey Web Button. Web Menu Creator

DBpedia Spotlight would return:

{ URI: "http://dbpedia.org/resource/Button_%28computing%29",
surfaceForm: "Button",
offset: 9,
similarityScore: 0.19474594295024872},

{ URI: "http://dbpedia.org/resource/Menu",
surfaceForm: "Menu",
offset: 21,
similarityScore: 0.13738319277763367}

Other additional information is returned, but not used in this paper.

In addition, we can specify a confidence value when querying the API, which will affect the number of entities the API returns. The larger the number the fewer the entities are returned. We use DBpedia data dump[5] to convert the DBpedia URIs to Freebase Ids (there is a one-to-one mapping), and eliminate those not exist in the entity.tsv provided by the ERD2014 organizers.

Afterwards, we eliminate all the unwanted surface forms specified in Section 3.3. And then we further filter the results using different thresholds for similarity score, which will be described in the evaluation section.

Afterwards, we add all non-overlapping entities from DBpedia to our result. For example, if originally we have the following entity:

Raleigh (104, 111) /m/0fvyg

and DBpedia proposes:

Raleigh, N.C. (104, 117) /m/0fvyg
Seattle (143, 150) /m/0d9jr

then only

Seattle (143, 150) /m/0d9jr

is added because offset (104, 117) is overlapped with (104, 111) in our original result.

3.6.2 TAGME

TAGME would annotate the long text with page ids and titles in Wikepedia. For example, if the long text begins with:

Grey Web Button. Web Menu Creator

TAGME would return:

{"id":731893,"title":"Grey","start":0,"rho":"0.05000","end":4,
"spot":"Grey"},

{"id":11143745,"title":"Web button","start":5,"rho":"0.50000",
"end":15,"spot":"Web Button"}

{"id":406283,"title":"Menu (computing)","start":21,
"rho":"0.13081","end":25,"spot":"Menu"}

where rho is a confidence value described in more detail in the short track, "spot" denotes the surface form, and "start" and "end" define the offset.

We use the Wikipedia data dump to convert the page ids to Wikipedia links (there is a one-to-one mapping), and then match those links with entity.tsv. Totally we match 2,350,947 page ids → links → Freebase Id, while entity.tsv contains 2,351,157 entries. We then eliminate any entity returned by TAGME that cannot be matched to the entity.tsv entries provided by the ERD2014 organizers.

Afterwards, we eliminate all the unwanted surface forms specified in Section 3.3.

TAGME itself may return overlapping entries. For example, if the long text begins with:

first time

TAGME may return "first time" as an entity and "time" as another entity at the same time.

When we detect this, we eliminate the one with lower rho. We will explore different threshold for rho in Section 4.

Afterwards, we add all non-overlapping entities from TAGME to our result, which has already incorporate results from DBpedia as specified before.

4. EVALUATION IN THE LONG TRACK

All experiments in this section except the formal evaluation are done with the development set. The development set contains 101 documents, but only 51 documents are labeled, so that the other 50 documents are not counted in expected F1. The formal evaluation contains 100 documents.

Table 1 shows the performance of DBpedia Spotlight, TAGME, and NTUNLP without integrating DBpedia and TAGME on the development set. Here, the Basic NTUNLP system includes detecting mentions using dictionary (Section 3.2), eliminating unwanted surface forms (Section 3.3), and linking to entities with Freebase API (Section 3.4). In DBpedia Spotlight, we use 0.3 as a threshold for confidence and 0.25 as a threshold for similarityScore. Besides, we only keep those entities existing in entity.tsv, and eliminate unwanted surface forms as specified in

[5] http://downloads.dbpedia.org/3.9/en/freebase_links_en.nt.bz2

Section 3.3. In TAGME, we do not use a threshold for rho, and only keep those entities existing in entity.tsv. As TAGME would emit overlapping entities, we eliminate them as specified in Section 3.6.2. We can see that both systems perform poorer than the Basic NTUNLP system.

Table 1. Performance of DBpedia Spotlight, TAGME, and Basic NTUNLP

	Evaluation		
	Precision	Recall	Expected F1
DBpedia Spotlight	0.9600	0.0208	0.0407
TAGME	0.4354	0.6670	0.5269
Basic NTUNLP	0.7492	0.6193	0.6781
- length 1 entities	0.7508	0.6193	0.6787
- length <= 4 numbers	0.7508	0.6193	0.6787
- stop words	0.7722	0.6175	0.6863
- all of the above	0.7756	0.6175	**0.6875**

We compare the performance gain by different elimination strategies specified in Section 3.3. When removing length 1 entities, length<=4 numbers, and stop words, we obtain an expected F1 0.6875 with precision 0.7756 and recall 0.6175. We can see that the performance gain is very small. This is because Freebase API may return entities not in the target set, which would be eliminated. In this way, we already eliminate most wrongly detected cases, because it is unlikely a wrong surface form can have a matching entry as its most relevant result returned by Freebase API. In our prior study, we use Wikipedia page views to estimate the popularity of each entity and simply select the most popular one for each surface form. Stop word list can increase performance significantly in this case.

Table 2 shows the performance of integrating the basic NTUNLP with DBpedia Spotlight (Section 3.6.1). We make some experiments to compare the performance with different thresholds for confidence parameter in DBpedia Spotlight.

Table 2. Performance of integrating Basic NTUNLP and DBpedia Spotlight with different thresholds for confidence

	Evaluation		
confidence	Precision	Recall	Expected F1
0	0.5231	0.6973	0.5978
0.1	0.6675	0.6930	0.6800
0.2	0.7206	0.6800	**0.6997**
0.3	0.7760	0.6340	0.6979
0.4	0.7753	0.6253	0.6923

Because DBpedia Spotlight will return a similarityScore ranging from 0 to 1 for each returned entity, which denotes the likelihood of the link between the entity and the mention, we also make some experiments to compare the performance when eliminating mentions with similarityScore lower than the threshold. Table 3 lists the experimental results. Here confidence is set to 0.2.

Table 4 lists the performance of integrating Basic NTUNLP and TAGME (Section 3.6.2). We also make some experiments to compare the performance when eliminating mentions with rho lower than the threshold.

Table 3. Performance of integrating Basic NTUNLP and DBpedia Spotlight with confidence 0.2 and different thresholds for similarityScore

	Evaluation		
similarityScore	Precision	Recall	Expected F1
0	0.7206	0.6800	**0.6997**
0.05	0.7229	0.6765	0.6989
0.1	0.7547	0.6323	0.6881
0.15	0.7692	0.6245	0.6893
0.2	0.7743	0.6219	0.6898
0.25	0.7739	0.6175	0.6869
0.3	0.7739	0.6175	0.6869
0.35	0.7756	0.6175	0.6875
0.4	0.7756	0.6175	0.6875

Table 4. Performance of integrating Basic NTUNLP and TAGME with different thresholds for rho

	Evaluation		
rho	Precision	Recall	Expected F1
0	0.4614	0.7832	0.5807
0.1	0.6586	0.7563	0.7041
0.2	0.7168	0.7112	**0.7140**
0.3	0.7471	0.6791	0.7115
0.4	0.7557	0.6574	0.7032
0.5	0.7705	0.6349	0.6961
0.6	0.7762	0.6227	0.6910
0.7	0.7758	0.6184	0.6882
0.8	0.7756	0.6175	0.6875
0.9	0.7756	0.6175	0.6875

When combining the Basic NTUNLP, DBpedia Spotlight, and TAGME all together, where we set confidence threshold to 0.2, similarityScore threshold to 0.0, and rho threshold to 0.3, we obtain an expected F1 **0.7173** with precision 0.7173 and recall 0.7173 under the development set.

Finally, after the organizers released the reference set on May 31, we compare our results and the labels in the reference set, and select 60 additional stop words which seem to be common words but produced by our system. This increases the performance to expected F1 **0.7410** with precision 0.7665 and recall 0.7173. But this result is just a reference, since we observe the ground truth.

The final NTUNLP system we use in the formal run is composed of all the methods specified in Sections 3.1~3.6. The thresholds for DBpedia Spotlight are confidence: 0.2, and similarityScore: 0.0; the threshold for TAGME is rho: 0.3. Besides, the stop word list includes the 60 stop words we found after examining the reference labeled data. In total, the list consists of 3,093 stop words.

Table 5 summarizes the results of dry run and formal run. When tested with the formal dataset, our system achieves similar performance as that with the development set. It shows the generality and robustness of our system. Indeed, as our system mostly uses a dictionary-based approach to detect surface forms and simply links them to the most relevant entities. It does not use any techniques specifically tuned for a given domain, so that we can expect similar performance across different data sets.

Table 5. NTUNLP in dry run and formal run of the long track

	Evaluation		
runs	Precision	Recall	Expected F1
Dry run	0.7173	0.7173	0.7173
Dry run+60	0.7665	0.7173	0.7410
Formal run	0.7571	0.6750	0.7137

5. SHORT TRACK

5.1 The Alias Dictionary

In order to solve the alias problem in the surface form, we firstly build a synonym dictionary (please refer to Section 3.1 for more detail) as controlled vocabulary. Different from the long track, we remove all the punctuations in the vocabulary and convert every surface form to lower case in the short track. There are two reasons for this preprocessing procedure: (1) normalizing the controlled vocabulary for string matching, and (2) the test queries given by ERD2014 server are in the lower case. Consider the Freebase webpage titled Britt Langlie as an example. The formatted entries of the example are shown as follows:

/m/05c3rc7 britt langlie
/m/05c3rc7 britt langli

In this example, /m/05c3rc7 is the Freebase Id, "britt langlie" is the Freebase title, and "britt langli" is an alias of "britt langlie".

5.2 Query Segmentation

In the short track, we firstly segment the given short text query into surface form candidates, and then try to find the candidates in the controlled vocabulary using string matching algorithm. Intuitively, if the length of a surface form is longer, then this surface form is more restrictive. For example, the surface form "american music award 2012" is longer than "american music award", so that the former is mapped to a smaller entity set. In our implementation, if both surface forms in the example can be found in the controlled vocabulary, our system will select "american music award 2012" instead of "american music award". In other words, we do the segmentation greedily: scan a query from the start to the end and try to match the longest surface form in the query to the controlled vocabulary. Substring matching is not used in our system because our preliminary experiment using substring matching shows a huge performance drop on the precision, although the recall is slightly increased. The longest surface form is regarded as a stable mention, and the segmentation process continues for the remaining string.

Below is a segmentation process for the short text query "chris brown nicki minaj"

Step 1: search "chris" in the controlled vocabulary, and result in:

/m/016fjj chris
/m/016nws chris
/m/02x706n chris
…, and the other 59 candidates.

Step 2: search "chris brown" in the controlled vocabulary, and result in:

/m/04hlp3 chris brown
/m/04m7my chris brown
/m/0h64616 chris brown
/m/07ss8_ chris brown
…, and the other 21 candidates.

Step 3: search "chris brown nicki" in the controlled vocabulary, and result in:

No candidate was found in the controlled vocabulary

Step 4: search "chris brown nicki minaj" in the controlled vocabulary, and result in:

No candidate was found in the controlled vocabulary

Step 5: reach to the end of the short text query. Our system saves "chris brown" as a stable mention, which has the following candidates:

/m/04hlp3 chris brown
/m/04m7my chris brown
/m/0h64616 chris brown
/m/07ss8_ chris brown
…, and the other 21 candidates.

Step 6: search "nicki" in the controlled vocabulary, and result in:

/m/03m45dg nicki
/m/06zmhjd nicki

Step 7: search "nicki minaj" in the controlled vocabulary, and result in:

/m/047sxrj nicki minaj

Step 8: reach to the end of the short text query. Our system saves "nicki minaj" as a stable mention, which has the following candidate:

/m/047sxrj nicki minaj

After scanning, we segment the query into two stable mentions, and report two candidate lists as follows:

"chris brown", which has:

/m/04hlp3 chris brown
/m/04m7my chris brown
/m/0h64616 chris brown
/m/07ss8_ chris brown
…, and the other 21 candidates.

"nicki minaj", which has the following candidate list:

/m/047sxrj nicki minaj

5.3 Candidate Selection

After the segmentation process, we got the mentions and a list of candidate(s) for each mention. The next step is to disambiguate the candidates. If there is no proper candidate, discard this mention.

Finding mentions and their corresponding candidates in the above step is relatively easier because it is just a string matching problem under the controlled vocabulary. Comparatively, to decide whether to discard or to keep the candidates is challenging. We often have no clues to make a decision. The following subsections describe our solution to this problem.

5.3.1 Stop Words Removal

In our observation, many proper nouns should be treated as the stop words, because (1) those proper nouns are rarely referenced by the crowd, and (2) those proper nouns have multiple meaning, and the most popular meaning is not a proper noun or an entity. Most of the not-so-good proper nouns are in the entertainment domain, such as song names, film names or novels. For example, in the query "entire process to obtain your real estate license", we find the mention "real estate" is also a name of a music band.

Because the amount of the ERD2014 test queries is very small and the query terms are very sparse, we build a list containing 3,033 stop words, which is the same stop word list used in the long track, to filter out the unwanted mentions.

5.3.2 Popularity-Based Selection

A simple and intuitive approach to select the best candidate from a candidate list for a mention relies on the use frequency of the entities. We postulate that a common-used entity is more likely to be our best candidate for a mention. Thus, the popularity-based selection method is to pick the most popular candidate in the candidate list. Here, we define the popularity of a candidate by the page view of its corresponding Wikipedia page in the most recent month.

Take the mention "chris brown" as an example. It has the following candidates along with their page views.

/m/04hlp3	chris brown	page view: 277
/m/04m7my	chris brown	page view: 288
/m/0h64616	chris brown	page view: 114
/m/07ss8_	chris brown	page view: 1,233
…, and the other 21 candidates		

Page view in this example is accumulated from Wikipedia within 2014/6/1-2014/6/26. All the other candidates in the list has page view lower than the candidate "/m/07ss8_ chris brown", which has 1,233 page view in the accumulation period. We can easily find the candidate "/m/07ss8_" is more popular than the others. Thus, our system will emit "/m/07ss8_ chris brown" as the result.

5.3.3 Utilizing Outside Resources

Before introducing the integration of our basic NTUNLP system and the outside resource TAGME, let us show an intriguing query we found in the development dataset. For the query text "youtube destinys child soldier", our segmentation process finds three mentions "youtube", "destinys child" and "soldier" together with their own candidate lists. In this example, we should return "youtube" and "destinys child", and discard "soldier", because there is no candidate link to the song "soldier" composed by the "destinys child" in the controlled vocabulary. Note that there are other "soldier" in the controlled vocabulary. "youtube" has no relation to "destinys child", but we should keep the result because there is a correct "youtube". Our solution to this problem is to utilize the outside resource.

Several systems can be explored to disambiguate the candidates. They are built on the huge knowledge database. Our first try is to use the Google search API, but we can only find the Wikipedia title "Soldier (Destiny's Child song)" with Google search API. Later on, we find the *TAGME* system, which is more suitable for our need. TAGME harvests the Wikipedia page links by comparison, and then links the mention in the given text to the corresponding Wikipedia page as annotation. Since each mention only links to at most one Wikipedia page, TAGME only returns one set of mention/Wikipedia page output.

Another outside resource is the Freebase. For each Freebase page, there is a direct hyper URL link that leads to its corresponding Wikipedia page. We then extract the links and Freebase Ids from the controlled vocabulary and build a Freebase Id to Wikipedia page pair table. We simply make an intersection between the candidate's Wikipedia links and the TAGME results based on this page pair table. The reason why we make an intersection between our segmentation result and TAGME result is the segmentation under a controlled vocabulary can improve the correctness of TAGME result. Here is an example of short text query "youtube destinys child soldier"

TAGME's result[6]:

youtube	/wiki/YouTube
destinys child	/wiki/Destiny%27s_Child
soldier	/wiki/Military_use_of_children

Our segmentation result:

youtube	/m/09jcvs	/wiki/YouTube
destinys child	/m/016ppr	/wiki/Destiny%27s_Child
soldier	/m/036g88	/wiki/Soldier_%281998_American_film%29
soldier	/m/0c3hzc	/wiki/Soldier_%281998_Indian_film%29
soldier	/m/0dx3zh	/wiki/Snake_Eater_%28film%29
soldier	/m/0t1np	/wiki/Soldier,_Iowa
soldier	/m/0t7g5	/wiki/Soldier,_Kansas

After the intersection process, we get the following result:

youtube	/m/09jcvs	/wiki/YouTube
destinys child	/m/016ppr	/wiki/Destiny%27s_Child

We can see that the mention "soldier" is discarded because there is no candidate left after the intersection between our segmentation result and the TAGME result. If there remain multiple candidates in a mention's candidates list after the intersection process, we select the one with the highest ρ (a score for a candidate annotation returned by the TAGME system) as the intersection result, so our system returns only one candidate for one mention.

5.3.4 Utilizing the Wikipedia

To further improve the performance, an intuitive idea is to give the candidate entities with higher scores if the candidate entities in the same query have some relations between them. To find the relation between two candidate entities, Wikipedia page would be a reliable resource. The following subsections introduce the *wiki title relation* and the *wiki link relation* we use in our system. The *wiki title relation* means the string of the mention can be found in the Wikipedia title, while the *wiki link relation* means there is a URL link between two candidates' Wikipedia pages. Among the 500 queries in the development dataset, there are 8 queries that have *wiki title relation*, and 43 queries that have *wiki link relation*.

5.3.4.1 Wikipedia Title Relation

After the candidate extraction, some words may not belong to any mention. For example, in the query "abc tv full episodes", we can only extract the mention "abc" from the controlled vocabulary, but the remaining terms "tv", "full" and "episodes" do not belong to any mentions, i.e., they cannot be matched with the controlled vocabulary or filtered out by the stop word list. We called those remaining words the *context words*, which may be a clue for the disambiguation phase.

If a candidate entity's Wikipedia title contains other mentions or context words (as a substring), we regard this entity as the best candidate of this mention. In practice, we do not do substring matching of the mentions or context words of length less than 4 to the Wikipedia title by our experience. If the mention/context word is too short, we assume the mention/context word has no relation

[6] /wiki/aaa is the abbreviation of "http://en.wikipedia.org/wiki/aaa"

to all candidate entities' Wikipedia titles in this given query. If our system matches a mention or a context word to a candidate's Wikipedia title, then this candidate's output priority is higher than the TAGME's result. For example, if the short text query is "berkeley square mini series", then our segmentation process results in:

Mentions		
berkeley square	/m/012dn_	/wiki/Berkeley_Square_%28TV_series%29
berkeley square	/m/0246k6	/wiki/Berkeley_Square
berkeley square	/m/026rf7s	/wiki/Berkeley_Square_%28play%29
Context word		
mini	(filtered out by stop word list)	
series	(filtered out by stop word list)	

and TAGME results in:

berkeley square	/wiki/Berkeley_Square

Since we can find the context word "series" in the Wikipedia title "/wiki/Berkeley_Square_%28TV_series%29", our wiki title relation results in:

berkeley square	/m/012dn_	/wiki/Berkeley_Square_%28TV_series%29

Our system takes "/wiki/Berkeley_Square_%28TV_series%29" as the output prior to TAGME's result "/wiki/Berkeley_Square". If a mention is discarded in the intersection phase, but found in the wiki title relation's result, our system will put the mention back because the priority of the wiki title relation is higher. If neither the mention nor the context word has wiki title relation between their candidates, we keep the intersection result derived in Section 5.3.3. If a given mention has multiple candidates that have the wiki title relation, then our system will return all of the candidates. This is the one and the only condition that our system returns multiple answer sets. However, the ERD2014 testing server provides a very sparse data set as the given context queries, only few queries among the 500 development set queries has wiki link or wiki title relations in them.

5.3.4.2 Wikipedia Link Relation

Intuitively, if there is a hyper link URL bridges between two candidate's wiki pages, they must have some relations between them. That is to say, they are very possible to be the best candidates for their corresponding mentions.

Every time we find a link between two candidates' Wikipedia pages, we will save these two candidates into our output result prior to the TAGME result. The detail procedure is similar to the Wikipedia title relation in the previous section. If there are multiple candidates containing wiki link relation in a candidate list of a given mention, our system will return all of them.

6. EVALUATION IN THE SHORT TRACK

6.1 Evaluation with the Development Set

The official development query set is provided by the organizers of ERD2014 workshop for dry run to test the system performance of each team. The development set consists of 500 different queries, each independent of the others. A query contains a short text whose average length is about 5 words. The following evaluation and result analysis is based on the development set. Finally, we will show the evaluation of our system in the formal run.

The basic NTUNLP system in the short track is composed of query segmentation (Section 5.2) and simple candidate selection (Section 5.3.1). Table 6 shows the expected F1 of two baselines (DBpedia Spotlight and TAGME) and the integration of the Basic NTUNLP system and TAGME. The confidence>0.2 and the similarity score>0 are two parameters in the DBpedia Spotlight system. The TAGME system will return a score ρ. If the returned ρ is less than the threshold value ρ_{NA}, then the corresponding TAGME's output will be discarded. In Table 6, $\rho_{NA} = 0$ means we keep all the TAGME' output. The notations "+stop word" and "−stop word" mean to use and not to use the stop word list to filter out the unwanted mentions, respectively.

The first two rows show that the TAGME system outperforms the DBpedia Spotlight system in the short track. The third row specifies that the simple popularity-based selection (Section 5.3.2) is useful, but its performance is 1.89% lower than that of TAGME. The 2nd-4th rows show that the integration of the Basic NTUNLP system and TAGME is better than both Basic NTUNLP + popularity and TAGME only. The last two rows show that the stop word list can increase performance.

Table 6. The expected F-score of integrating the Basic NTUNLP and TAGME with the short track development set

method	Evaluation Expected F1
1. DBpedia (confidence > 0.2, similarity score > 0), +stop words	0.4600
2. TAGME ($\rho_{NA} = 0$), +stop words	0.5855
3. Basic NTUNLP, +popularity, −stop words	0.5666
4. Basic NTUNLP+TAGME ($\rho_{NA} = 0$), −stop words	0.6201
5. Basic NTUNLP+TAGME ($\rho_{NA} = 0$), +stop words	**0.6321**

6.2 Parameter Sensitivity of ρ_{NA}

The parameter ρ in the TAGME system estimates the "goodness" of the annotation with respect to the topics of the input text. In the previous experiments, we set $\rho_{NA} = 0$ for all methods. However, the threshold can be chosen within the interval [0,1]. Table 7 shows the result using TAGME as the outside resource and filtered out the unwanted candidates by the stop word list with different ρ_{NA}. Surprisingly, the best ρ in this experiment is the same as the TAGME's official suggestion. As a result, we set ρ_{NA}=0.1 in the following experiments.

Table 7. The expected F-score of integrating the Basic NTUNLP and TAGME with different thresholds

ρ_{NA}	Evaluation Expected F1
0	0.6321
0.08	0.6601
0.1	**0.6621**
0.12	0.6541
0.2	0.6373
0.3	0.6060

6.3 Utilizing the Wikipedia Relations

Two additional candidate selection methods, i.e., (1) plus the wiki title relation (Section 5.3.4.1), and (2) plus the wiki link relation (Section 5.3.4.2), are introduced. The results are shown in Table 8.

Since the performance improvement of the wiki title relation method is not as good as our expectation, we further investigate the 8 additional queries. Some queries turns from right to wrong. A possible explanation of this outcome is that the Wikipedia clues are not very reliable. Although the performance gain of "+wiki title relation" seems to be just a little in this development dataset, we still hope that this method can help us gain some improvement in the ERD2014 formal run.

The performance change of wiki link relation does not meet our expectation. We originally think the wiki links might be a strong evidence between two candidates that are likely to be the correct answers, but the experimental result shows this conjecture may be wrong. The use of "+wiki link relation" even decreases the performance by 0.012. That is to say, the Wikipedia link is useless. We decide not to implement this in our final system.

Table 8. The expected F-score of integrating the Basic NTUNLP, TAGME and Wikipedia information

method	Evaluation Expected F1
1. Basic NTUNLP+TAGME ($\rho_{NA} = 0.1$), +stop words	0.6621
2. Basic NTUNLP+TAGME ($\rho_{NA} = 0.1$) + stop words+wiki title relation	**0.6641**
3. Basic NTUNLP+TAGME ($\rho_{NA} = 0.1$) + stop words+ +wiki link relation	0.6501

6.4 Evaluation in the Formal Run

There are 500 queries in the formal run query set. We adopt the best model in the dry run, i.e., Basic NTUNLP+TAGME ($\rho_{NA} = 0.1$) +stop words+wiki title relation. The expected F1 score of the model in test query set is 0.6797, which is a little better than the performance in the development set. It shows the robustness of our system, because we do not over-train our system to achieve the incredible high performance in development query set. The general method can perform reasonable performance on the unknown query dataset.

7. CONCLUSION AND FUTURE WORKS

In this paper, we introduce the NTUNLP approaches to recognizing and disambiguating entities appearing in the long and in the short text. In the 2014 ERD challenge, our NTUNLP system is the winner of the first prize in the short track. The expected F1 score of the NTUNLP system ranks the 4th places in the long track.

In the short track, there are still many places to improve the performance of the NTUNLP system. Currently, we return multiple answer set in only such a case that multiple candidates contain wiki title relations in a candidate list of a mention. We can further enhance our system to return the best N candidates or return all candidates exceeding a predefined threshold.

In the long track, the basic NTUNLP system adopts a dictionary-based approach to detect the possible surface forms and relies on Freebase API to link to the most relevant entity. This relevance is mostly about the popularity of entities and the possibility of a given surface form linked to different entities. That is, it does not use any contextual information for disambiguation. We explore the integration of the basic NTUNLP with DBpedia Spotlight and TAGME, which do use context information to help disambiguation. Because we only include non-overlapping entities to our result, the final output still relies most on our original system. As we learn from the short track, contextual information and relation between different entities would be helpful in this task. The precision of our system can still be improved by using context information for disambiguation.

8. ACKNOWLEDGMENTS

This research was partially supported by the Project D352B23100 (Industrial Technology Research Institute) under the sponsorship of the Ministry of Economic Affairs, Taiwan, R.O.C., and the grant 103R890858 by National Taiwan University.

9. REFERENCES

[1] D. Carmel, M.W. Chang, E. Gabrilovich, B.J. Hsu, and K. Wang, "ERD 2014: Entity Recognition and Disambiguation Challenge," *SIGIR Forum*, 2014 (forthcoming), ACM.

[2] P. N. Mendes, M. Jakob, A. García-Silva, and C. Bizer, "DBpedia spotlight: shedding light on the web of documents," In *Proceedings of the 7th International Conference on Semantic Systems*, 2011, pp. 1–8.

[3] P. Ferragina and U. Scaiella, "Tagme: on-the-fly annotation of short text fragments (by wikipedia entities)," In *Proceedings of the 19th ACM international Conference on Information and Knowledge Management*, 2010, pp. 1625–1628.

[4] R. Mihalcea and A. Csomai, "Wikify!: linking documents to encyclopedic knowledge," In *Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management*, 2007, pp. 233–242.

[5] D. Milne and I. H. Witten, "Learning to link with wikipedia," In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, 2008, pp. 509–518.

[6] G. Salton, A. Wong, and C.-S. Yang, "A vector space model for automatic indexing," *Commun. ACM*, vol. 18, no. 11, pp. 613–620, 1975.

[7] S. Hakimov, H. Tunc, M. Akimaliev, and E. Dogdu, "Semantic question answering system over linked data using relational patterns," In *Proceedings of the Joint EDBT/ICDT 2013 Workshops*, 2013, pp. 83–88.

[8] A. V. Aho and M. J. Corasick, "Efficient string matching: an aid to bibliographic search," *Commun. ACM*, vol. 18, no. 6, pp. 333–340, 1975.

[9] S. Hakimov, S. A. Oto, and E. Dogdu, "Named entity recognition and disambiguation using linked data and graph-based centrality scoring," In *Proceedings of the 4th International Workshop on Semantic Web Information Management*, 2012, p. 4.

[10] A. Fader, S. Soderland, O. Etzioni, and T. Center, "Scaling Wikipedia-based named entity disambiguation to arbitrary web text," In *Proceedings of the IJCAI Workshop on User-contributed Knowledge and Artificial Intelligence: An Evolving Synergy, Pasadena, CA, USA*, 2009, pp. 21–26.

[11] R. C. Bunescu and M. Pasca, "Using Encyclopedic Knowledge for Named entity Disambiguation," In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL-06)*, 2006, vol. 6, pp. 9–16.

[12] S. Cucerzan, "Large-Scale Named Entity Disambiguation Based on Wikipedia Data," In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language*

Processing and Computational Natural Language Learning, 2007, vol. 7, pp. 708–716.

[13] C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann, "DBpedia-A crystallization point for the Web of Data," *Web Semant. Sci. Serv. Agents World Wide Web*, vol. 7, no. 3, pp. 154–165, 2009.

[14] Google, *Freebase Data Dumps*, March 23, 2014. 2014.

[15] P. Ferragina and U. Scaiella, "Fast and accurate annotation of short texts with Wikipedia pages," *ArXiv Prepr. ArXiv10063498*, 2010.