FYP Final Report
Entity Resolution via Canonicalization

FYP18002
Feng Xiyang
UID: 3035235317

April, 2019
Abstract

Nowadays, open knowledge base has been a research topic of much interest. Open knowledge base usually serves as the fundamental data layer of many Natural Language Processing tasks and significantly affects their performance. However, large open knowledge base usually suffer from low accuracy and high redundancy which requires further canonicalization. Canonicalization is the process of determining whether two mentions having the same identity. This project presents a state of the art canonicalization system called MLCE which utilizes the embedding method. This report reviews some current open KB canonicalization systems such as GHMS and CESI. It also discusses the detailed procedure as well as the rational. Finally, the experimental setup, results and corresponding analysis are presented at the end of this report.
# Table of Contents

1 Introduction
   1.1 Backgrounds ......................................................... 1
   1.2 Scope .............................................................. 2
   1.3 Objective .......................................................... 2
   1.4 Outline ............................................................. 2

2 Related Studies
   2.1 GHMS ................................................................. 3
   2.2 GloVe and Word Embedding ........................................... 4
   2.3 CESI ................................................................. 5
   2.4 BERT and Sentence Embedding ....................................... 6

3 Proposed Approach: MLCE ............................................. 7

4 Word Level Canonicalization ........................................... 9
   4.1 Motivation .......................................................... 9
   4.2 Procedure ........................................................ 10

5 Sentence Level Canonicalization ...................................... 12
   5.1 Motivation ........................................................ 12
   5.2 Procedure ........................................................ 13

6 Experimental Setup ..................................................... 16
   6.1 Datasets .......................................................... 16
   6.2 Data Cleaning ...................................................... 18
   6.3 Evaluation Metrics ................................................ 19
      6.3.1 Macro ......................................................... 20
      6.3.2 Micro ........................................................ 20
      6.3.3 Pairwise ...................................................... 20
      6.3.4 Canonicalization Evaluation Example ....................... 21
6.4 Methods Compared .................................................. 22

7 Results ........................................................................ 23
  7.1 Results and Analysis .................................................. 23
  7.2 Case Study .............................................................. 26
    7.2.1 Mumbai and Bombay Case ..................................... 26
    7.2.2 Airport Case ...................................................... 28
  7.3 Difficulties and Future Work ....................................... 28
    7.3.1 Blocking Criteria ................................................ 28
    7.3.2 Apple Problem ................................................... 29

8 Conclusion .................................................................... 30

9 Contribution .................................................................. 31

References ..................................................................... 32
List of Figures

1  GHMS Workflow .................................................. 3
2  Word Embedding in 3D Space ................................. 5
3  CESI Workflow .................................................... 6
4  BERT Structure and Results ................................. 7
5  MLCE Workflow .................................................. 8
6  Mumbai Cluster with GHMS ................................. 10
7  Bombay Cluster with GHMS ................................. 10
8  Detailed Workflow of Word Level Canonicalization ...... 11
9  Hong Kong Cluster in Level 1 Canopies ................. 13
10 Taipei Cluster in Level 1 Canopies ..................... 13
11 Detailed Workflow of Sentence Level Canonicalization ... 14
12 Capture of Ground Truth Clusters in Reverb Freebase dataset ... 17
13 Illustrative Example for Evaluation Metrics ............. 21
14 Mumbai Cluster with CESI ................................. 26
15 Mumbai Cluster with MLCE ................................. 27
16 Hong Kong and Macau Clusters with MLCE ........... 27
17 Taipei and Kaoshiung Cluster with MLCE ............. 27
18 Airport Clusters with CESI ................................. 28
19 Airport Clusters with MLCE ................................. 28
20 Fine-tuning Designs ........................................... 29

List of Tables

1  Comparison between Curated KBs and Open KBs ........ 1
2  Meta Data for Datasets Used .................................. 16
3  Metrics Score for the Illustrative Example ................ 21
4  Canonicalization Result for Ambiguous Dataset ....... 23
5 Canonicalization Result for Reverb Freebase Dataset . . . . . . . . 24
6 Canonicalization Result for Reverb Wiki Dataset . . . . . . . . . . 24
7 Canonicalization Result for Reverb Intersect Dataset . . . . . . . . 24
8 Canonicalization Result for Reverb Unioni Dataset . . . . . . . . . 25
1 Introduction

1.1 Backgrounds

The term knowledge base (KB) refers to a system designed for storing complex structured and unstructured information in computer science. Knowledge base usually serves as the fundamental data layer for a number of tasks such as question answering and language inference. Therefore, the research of knowledge base has long been a topic of much interest.

Existing knowledge bases are either curated KB or open KB. Curated KBs, whose information is modeled by the resource description framework, are manually examined and calibrated. Open KBs, on the other hand, are essentially a collection of assertions from online text resources [1]. These assertions are usually stored in a triple form of (subject, relation, object). For example, (Barack Obama, is husband of, Michelle Obama) is a sample assertion in open KBs where is husband of represents the relation.

<table>
<thead>
<tr>
<th></th>
<th>Curated KBs</th>
<th>Open KBs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Completeness</td>
<td>Incomplete</td>
<td>Comprehensive</td>
</tr>
<tr>
<td>Accuracy</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>Redundancy</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td>Search Speed</td>
<td>High</td>
<td>Low</td>
</tr>
</tbody>
</table>

Table 1: Comparison between Curated KBs and Open KBs

Table 1 shows a direct comparison between these two types of KBs. Curated KBs, with human calibration, have lower redundancy, higher accuracy and higher speed, and thus are more suitable for tasks that require instant and precise response from knowledge base. However, the incompleteness of knowledge deems that curated KBs can only cover a small portion of information which is insufficient for many tasks [2]. And, unfortunately, the manual effort involved in calibration makes curated KBs extremely difficult to scale.
Hoping to take advantage of the comprehensive knowledge, some researchers turn to open KBs and work on canonicalization. Canonicalization is the task to disambiguate identity of entities in KBs. Since open KBs require no manual examination, a single entity may have different representations. For example, the entity *Barack Obama* may appear also as *President Obama* or *Husband of Michelle Obama* in open KBs. The process of clustering different representations with the same meaning is an example of canonicalization [3] which will make the knowledge base more accurate and compressed.

### 1.2 Scope

The area of canonicalization generally falls into two categories - noun phrases canonicalization and verbal phrases canonicalization which focus on canonicalizing subjects/objects and relations respectively. Given the difficulty of verbal phrases canonicalization and the time span of this final year project, this project focuses on noun phrases canonicalization of subject.

### 1.3 Objective

The objective of this project is to build a fast and accurate canonicalizer that can canonicalize any open KBs within a reasonable amount time. Our method can also solve a special case where entities with the same meaning may have completely different representations such as *Mumbai* and *Bombay*.

### 1.4 Outline

The remainder of this report proceeds as follows. First, we review some related works on both knowledge base canonicalization and embedding methods. Then we present our approach Multi-level Canonicalization with Embedding (MLCE) and its detailed
procedures. We then describe our experimental setup and corresponding results. We close with a summary of our project as well as future work.

2 Related Studies

In this section we will review two widely used open KB canonicalization systems. We will also introduce some recent language embedding methods and their state of art language models.

2.1 GHMS

A standard open KB canonicalization process usually involves two steps. The first step is to measure the similarity between two entities by a pairwise function such as *edit-distance function*. And the second step is to cluster entities by performing Hierarchical Agglomerative Clustering (HAC) algorithm which is still the most widely used clustering algorithm for canonicalization today.

<table>
<thead>
<tr>
<th>Assertion</th>
<th>Subject</th>
<th>Relation</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>Barack Obama</td>
<td>Grew up in</td>
<td>Honolulu</td>
</tr>
<tr>
<td>A2</td>
<td>Barack Obama</td>
<td>Is the president of</td>
<td>The US</td>
</tr>
<tr>
<td>A3</td>
<td>President Obama</td>
<td>Was born in</td>
<td>1962</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

![GHMS Workflow](image)

Figure 1: GHMS Workflow
Before the emergence of embedding method, traditional canonicalization systems usually evaluate entity similarity over some manually defined features. In 2015, Galarraga proposed a canonicalization called GHMS and performed a comprehensive test on many different similarity functions [4]. Figure 1 shows the main flow of GHMS method. In blocking step, GHMS method assigns entities to different canopies based on the tokens of the entity. It then runs a clustering algorithm inside each canopy and obtains a number of sub-clusters in canopies. Finally, GHMS merges sub-clusters that contain overlapped entities and get the final result. Such blocking-clustering-merging design significantly reduce the problem size that allows the clustering algorithm to finish in a reasonable amount of time.

Galarraga claimed that among all the similarity functions tested, the Inverse Document Frequency (IDF) Token Overlap is the one that yields the best result. IDF Token Overlap similarity is essentially a string comparison but with each token carrying a weight equals to its inverse document frequency. This method works well in most cases. However, it has a major drawback in dealing with the case where two entities share the same meaning but have completely different representations such as Mumbai and Bombay. These two entities have no overlapped tokens, so IDF method will never consider them as the same entity. Therefore, a more meaningful similarity function that considers not only string similarity but also contextual information is needed.

2.2 GloVe and Word Embedding

Word Embedding [5] is a new method of word representation that emerges recently. It maps words to high dimensional space vectors and measures words similarity by their cosine distances. Traditional word representation methods like one-hot vector representation fail to consider the contextual and syntactic information. For example, word pair (dog,cat) is considered equally different as pair (dog,car) in one-hot vector representation. However, dog and cat should have more common attributes in our knowledge.
Such problem is solved by the Word Embedding method.

Figure 2 shows the Word Embedding representations for word *cat*, *dog* and *car* in the three dimensional space. The shorter distance between *dog* and *cat* indicates a higher similarity compared with the distance between *dog* and *car*. As a result, the semantic and syntactic information are included in the Word Embedding representation.

Global Vectors for Word Representation (GloVe) is a pre-trained word vector model that contains about 6 billion tokens and 400 thousand vocabularies with an embedding dimension of 300 [6]. This model is trained on Wikipedia data corpus which covers almost all the vocabularies needed for open knowledge base canonicalization.

### 2.3 CESI

In 2018, Vashishth proposed a new canonicalization system called CESI (Canonicalization using Embeddings and Side Information) which is one of the first canonicalization systems that utilizes the embedding method [7]. Figure 3 shows the work flow of CESI method. Instead of measuring similarity by string comparison, CESI maps entities to a high-dimensional space by assigning a vector representation of the entity based on GloVe model and side information. It then uses HAC algorithm to identify clusters in
the hyper-space.

Apart from side information, the meat of CESI lies in the use of word embedding model to obtain a vector representation which encodes not only string similarity but also contextual information. As a result, CESI method can successfully handle the Mumbai and Bombay case. However, by using word embedding, CESI method tends to put entities with similar meanings rather than same meanings into a cluster. Therefore, further divisive step is needed for CESI method in order to achieve the optimal outcome.

2.4 BERT and Sentence Embedding

Shortly after the emergence of word embedding [5], Mikolov proposed the idea of sentence embedding in 2014 [8]. Similar to word embedding, sentence embedding maps every token in the sentence to a high dimensional space and assign them a vector representation. Depending on the circumstances, sometimes an extra token may be added to encode the whole sentence. In most of the time, sentence embedding, which also encodes the position, occurrence and other properties of the token inside a sentence, can be considered as an extended version of word embedding and can be modified accordingly for specific tasks.
BERT [9] is a pre-trained language model that achieves state of the art performance for many NLP tasks (see Figure 4). BERT model implements the concept of transfer learning. Transfer learning is a learning strategy that learns a new task through the transfer of knowledge from a related task that has already been learned. This strategy is usually used when the knowledge of two tasks are readily transferable and the training data is hard to acquire. In the BERT case, the model learns the general knowledge of vocabularies and can be fine tuned for specific tasks.

3 Proposed Approach: MLCE

Given a pair of entities in an open knowledge base, their relations generally falls into four categories as listed below.

- **Case 1:** same meaning and similar (or same) representations.
  
  Example: *Barack Obama* and *President Obama*

- **Case 2:** same meaning but different representations.
  
  Example: *Mumbai* and *Bombay*

- **Case 3:** different meaning and different representations.
Example: *Barack Obama* and *Mumbai*

- Case 4: different meaning but similar representations.
  Example: *Barack Obama* and *Michelle Obama*

- Case 5 (special and difficult): different meaning but same representations.
  Example: *Apple (Company)* and *Apple (Fruit)*

To our best knowledge, without considering the special case 5, existing canonicalization systems can effectively handle at most 3 cases among the first 4 listed above and fail on either case 2 (GHMS method) or case 4 (CESI method). To tackle the four cases, we propose a new approach for open KB canonicalization called MLCE (Multi-level Canonicalization With Embedding) which involves two major canonicalization steps named word level canonicalization and sentence level canonicalization respectively.

Figure 5: MLCE Workflow

Figure 5 shows the work flow of MLCE method. The input to the system is an uncanonicalized Open KB. After word level canonicalization, we obtain the interim result named level 1 canopies. Level 1 canopies, which can also be viewed as categories, are
coarse clusters that contain entities with similar meanings. In Figure 5, the original knowledge base is divided into two categories - the category of Obama family and the category of Indian cities after the first step. MLCE then performs a sentence level classification and obtains the final result. The output of MLCE is a list of clusters which can be used to identify equivalent entities in the open knowledge base.

Details of two steps of MLCE are described in the next two sections.

4 Word Level Canonicalization

This section discusses the motivation, target problems as well as the detailed steps for word level canonicalization.

4.1 Motivation

The word level canonicalization step has two major targets. The first target is to enhance the calculation efficiency, i.e. to make the algorithm applicable to large open knowledge bases. Clustering algorithms, if not provided with the desired number of clusters, are usually computational costly with a time complexity on $O(n^3)$ order. Since it is impossible to predict the number of clusters in open KBs, the HAC algorithm has also to run in $O(n^3)$ time and $O(n^2)$ space. Considering canonicalizing an middle size open knowledge base with 1 million of assertions, the HAC algorithm will require at least $10^{18}$ times of execution and 900GB of memory which is quite impractical. Therefore, we need to find a way to divide the original problem into multiple smaller sub-problems.

Another motivation of using word level canonicalization is to handle Case 2 problem (entities with same meaning but different representations) as stated in Section 3. Considering the canonicalization of two entities Mumbai and Bombay, Recall that in GHMS method, Mumbai and Bombay will be assigned to different canopies in the blocking step. Because they share no overlapped tokens, their canopies will never get the change
to be merged together. As a result, Mumbai and Bombay will always end in different clusters in GHMS method.

Figure 6: Mumbai Cluster with GHMS

Figure 7: Bombay Cluster with GHMS

Figure 6 and figure 7 is the canonicalization result for Mumbai and Bombay in GHMS method. The fact that they belong to different clusters matches our analysis above. In order to handle this case, we need a better measurement of entity similarity that also considers the contextual information.

4.2 Procedure

The workflow of word level canonicalization is built upon the blocking - clustering - merging framework in GHMS method which widely used by nowadays canonicalization systems.
Figure 8 shows the detailed workflow of word level canonicalization (WLC). Given an un-canonicalized open knowledge base, WLC first builds a map $M$ that maps a subject to all assertions which contain the subject (blocking by subject step). For example in figure 8, $M$ will map Barack Obama to three different assertions. After obtaining map $M$, we collect all different subjects by collecting the keys of $M$. Since GloVe embedding only considers the subject, the relation and object in an assertion will become irrelevant information in this step. By only considering different subjects, we can significantly reduce the problem size and make it practical for HAC clustering. Assume the average number of assertions associated with a subject is $N$ ($N$ is usually greater than 20 for most open knowledge bases), then the blocking by subject step will reduce the problem by $N$ times which is a significant enhancement in terms of computation efficiency.

After obtaining the collection of different subjects, WLC then maps each subject to a 300-dimension space through GloVe embedding and performs HAC algorithm in this hyperspace. Finally, WLC expands each subject to its associated assertions and results in the level 1 canopies. In the example of figure 8, the Mumbai Bombay cluster is expanded to the right side level 1 canopy.
Note that each level 1 canopy can be regarded as a category. The range of a category depends on the threshold set in HAC algorithm. A smaller threshold tends to narrow down the category and a larger threshold tends to include more entities. In the MLCE method, we want the level 1 canopies satisfies two properties. The first property is that each level 1 canopy should have a reasonable size since another canonicalization step will be executed against them. For the second property, given two entities from two different level 1 canopies, these two entities should have different meanings, i.e. they should belong to different clusters in the ground truth. Because the MLCE method is a coarse-to-fine divisive approach that does not include any merging step, entities belong to different level 1 canopies will never get the chance to be clustered together in the remaining steps. As a result, we want the second property hold for as many entity pairs as possible. And this can be achieved by setting a relatively larger threshold for HAC algorithm.

5 Sentence Level Canonicalization

This section discusses the motivation, target problems as well as the detailed steps for sentence level canonicalization.

5.1 Motivation

In order to solve the Mumbai and Bombay problem, the word level canonicalization adopts GloVe word embedding and clusters entities with similar meanings together. However, in most of the time, similar meaning does not imply same identity.

Figures below are gathered from the result after word level canonicalization. Figure 9 is the Hong Kong cluster and figure 10 is the Taipei cluster in level 1 canopies. It is not surprising to see that both clusters are impure and mixed with other entities with similar meanings such as Macau in Hong Kong cluster and Kaoshiung in Taipei cluster.
As discussed in section 4.2, the nature of each level 1 canopy is a category. It is only guaranteed entities from different categories having different meaning but no promise on entities from the same category having the same meaning. Ideally, Hong Kong, Macau, Taipei and Kaoshiung should end up in different clusters in the final result. In order to achieve this, we need a further divisive method against level 1 canopies. And this is the motivation of using the sentence level canonicalization.

5.2 Procedure

Similar to word level canonicalization, the design of sentence level canonicalization also follows the blocking - clustering - merging framework with modifications specifically designed for this project.

Figure 11 shows the detailed workflow of sentence level canonicalization (SLC) for
two different cases. The input to sentence level canonicalization is the level 1 canopies obtained in word level canonicalization. In the first step, we make an assumption that entities with the same representation must have the same meaning. Given this assumption, if a level 1 canopy contains $N$ different entities, there are at most $N$ different
clusters in the final result. Therefore, we first assign assertions to temporary canopies based on their subject in the blocking step.

After blocking, SLC then uses bert embedding to obtain a sentence level vector representation for every assertion. In bert embedding, every token in the sentence will be assigned a vector representation which encodes both their contextual meaning and properties within the sentence. For example, considering an assertion (*Barack Obama, is president of, United States*) will be assigned 7 vectors each corresponding to a word. And the assertion vector is defined as the average of token vectors in the subject. Using the same example above, the assertion vector is given by \( \frac{V_{\text{barack}} + V_{\text{obama}}}{2} \).

With the assertion vectors, SLC further obtains an average vector for each temporary canopy. This can be easily calculated by averaging every assertion vectors within the temporary canopy. For example, in figure 11, the temporary canopy vector for *Barack Obama* is defined as \( V_{\text{barack--obama}} = \frac{V_1 + V_2 + V_4}{3} \) and *Mumbai* canopy vector is defined as \( V_{\text{mumbai}} = \frac{V_5 + V_7}{2} \). Apart from this temporary canopy vector design, the original SLC directly canonicalize every assertions after obtaining their assertion vectors. Base on many experiments and observations, the bert embedding is heavily influenced by the sentence structure which may create unnecessary splits if directly canonicalized. Considering an example of two assertions (*Barack Obama, is president of, United States*) and (*Barack Obama, visits, China*), the assertion vector obtained from bert embedding has a distance larger than expectation. This is understandable because the first assertion is about the occupation of a person and the second assertion is about the action of a person which shares very different sentence structure. Bert embedding essentially is a sentence embedding model which emphasizes a lot on sentence structure. However, this circumstances is unacceptable in canonicalization. In order to weaken the influence of sentence structure, we come up with the temporary canopy vector design. By averaging assertion vectors within a temporary canopy, each sentence will have minor effect on the canopy vector. And this the main reason of using temporary canopy vector
in SLC.

After obtaining the temporary canopy vectors, SLC then runs HAC algorithms to identify clusters of temporary canopies. Finally, SLC obtains final by expanding temporary canopies to their related assertions. Figure 11 shows an ideal case where Barack Obama canopy and Michelle Obama canopy are considered as different while Mumbai and Bombay canopies are considered as the same. And the final result matches the ground truth.

6 Experimental Setup

This section first discusses the datasets used and their pre-process before experiments. It then introduces three evaluation metrics for the quality of canonicalization result. Finally, this sections will talk about what other methods MLCE is going to compete with.

6.1 Datasets

Statistical data of the datasets used in this project is listed in Table 2. A summary of each dataset is presented below.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>#triples</th>
<th>#clusters</th>
<th>#singletons</th>
<th>largest cluster</th>
</tr>
</thead>
<tbody>
<tr>
<td>ambiguous</td>
<td>34209</td>
<td>446</td>
<td>0</td>
<td>3555</td>
</tr>
<tr>
<td>reverb freebase</td>
<td>36814</td>
<td>6061</td>
<td>0</td>
<td>569</td>
</tr>
<tr>
<td>reverb wiki</td>
<td>36814</td>
<td>7060</td>
<td>3146</td>
<td>580</td>
</tr>
<tr>
<td>reverb intersect</td>
<td>36814</td>
<td>8431</td>
<td>2642</td>
<td>558</td>
</tr>
<tr>
<td>reverb union</td>
<td>36814</td>
<td>5124</td>
<td>0</td>
<td>3579</td>
</tr>
</tbody>
</table>

Table 2: Meta Data for Datasets Used

- **Ambiguous dataset**: Ambiguous dataset is obtained from the author of GHMS [4]. This dataset is created by first collecting Freebase [10] entities that have
multiple representations with a number of 150. Ambiguous dataset is then constructed by enrichment assertions related to these entities. Table 2 shows Ambiguous dataset contains about 500 entities each of which is associated with an average of 70 assertions.

- **Reverb Freebase dataset**: Reverb Freebase dataset is obtained from the author of CESI [7]. The original dataset is called Reverb45k which is a significant extended version of Ambiguous dataset. The Reverb45k contains about 7500 entities with an average cluster size of 6. It is constructed by combining information from Reverb Open KB, Freebase and Clueweb09 wiki corpus. As a result, each entity in Reverb45k is linked to either a Freebase ID or a Wikipedia Link. The Reverb Freebase dataset is constructed from Reverb45k dataset by using Freebase ID as the ground truth. To sum up, the Reverb Freebase dataset is a more realistic dataset with extra entities, assertions as well as side information (such as source sentence) compared with Ambiguous dataset. Thus, this dataset is a harder problem for open KB canonicalization systems.

  ![Figure 12: Capture of Ground Truth Clusters in Reverb Freebase dataset](image)

- **Reverb Wiki dataset**: Reverb Wiki dataset is constructed from Reverb Freebase dataset by using Wiki Link as the ground truth. Based on observations, the ground truth of Reverb Freebase dataset can sometimes be confusing. Figure 12 shows one of the ground truth clusters in Reverb Freebase dataset. Obviously *African American*, *American* and *Michelle Obama* should belong to different ground truth clusters. Facing this problem, we create the Reverb Wiki dataset which uses Wiki Link as the ground truth. Wiki Link exists in about 80% of entities. For those without Wiki Link, MLCE will ignore them in evaluation. It is hard to
argue which ground truth is more suitable for evaluating canonicalization results in a systematic way. However, we can argue that a good canonicalization result should have a consistent good performance on both Reverb Freebase dataset and Reverb Wiki dataset. Therefore, it is reasonable to create Reverb Wiki dataset and include it in the evaluation.

- **Reverb Intersect dataset**: Reverb Intersect dataset is constructed by considering the intersection of Freebase ID and Wiki Link as the ground truth. Given two entities with Freebase ID and Wiki Link existed, these two entities have same ground truth if and only if they have the same Freebase ID and the same Wiki Link. This property holds for every entity pair in Reverb Intersect. If only one of Freebase ID and Wiki Link exists, the ground truth is based on the one present. If none of the them exists, the entity will be ignored during evaluation. In general, Reverb Intersect dataset is a divisive version of Reverb45k. Note that the quality of Reverb Intersect dataset may not be as good as Reverb45k since impure clusters are created when merging entities that only have Freebase ID or Wiki Link. Details about the quality of datasets will be evaluated in results section.

- **Reverb Union dataset**: Similar to Reverb Intersect dataset, Reverb Union dataset is constructed by considering the union of Freebase ID and Wiki Link. Given two entities with Freebase ID and Wiki Link existed, these two entities have the same ground truth if they have the same Freebase ID or the same Wiki Link. The Reverb Union dataset is a clustered version of Reverb45k. It also suffers from low quality when merging entities with only Freebase ID or Wiki Link. The largest cluster in Reverb Union dataset contains over 3000 entities which could significantly hurt the evaluation.

### 6.2 Data Cleaning

Recall that MLCE requires the level 1 canopies to satisfy the property that entities from different level 1 canopies have different meanings. This property relies heavily on the
GloVe embedding which fails when the pre-trained model has not seen an entity before. Fortunately, GloVe is trained on enormous data corpus and contains information of over 400 thousand vocabularies. However, it still cannot handle the typo situation. Consider a typo of *barack obama* as *barackobama*. The GloVe embedding will consider *barack-obama* as a single word and fails to find its vector representation. Such case has gone beyond the scope of open knowledge base canonicalization. Therefore, we need to fix the typo and clean the data before canonicalizing them. The cleaning step processes as follow.

First we define a term *ambiguous token* which refers to a token that appears in three or more than three different entities. Consider the token *Obama*, it is not a an *ambiguous token* if it only refers to entity *Obama* and *Barack Obama*. However, if it refers to a third entity *Michelle Obama*, we may regard it as an *ambiguous token*. Therefore, three is chosen as the threshold in defining *ambiguous token*. An entity is an *ambiguous entity* if it contains any *ambiguous token*.

Then for every pair of entities, if any of them is *ambiguous entity*, we skip this pair since solving ambiguous entity is in the scope of open knowledge base canonicalization. If none of them is *ambiguous entity* and their edit distance is smaller or equal to 2 which means one can be transformed to the other within 2 operations, then we consider one of them is a typo and assign them a unified representation. This process will be executed against every dataset before inputting to MLCE.

### 6.3 Evaluation Metrics

This section introduces three metrics for evaluation canonicalization results. A concrete evaluation example is presented at the end of this section. In this section, C denotes the canonicalization result to be evaluated and E denotes the actual canonicalization (also known as ground truth). F1 score is the harmonic mean of precision and recall.
6.3.1 Macro

Macro precision is fraction of pure clusters in $C$, i.e., clusters that are true subset of ground truth clusters [7]. Macro recall is defined the same as macro precision except for the exchange of role of $C$ and $E$.

$$P_{macro}(C, E) = \frac{|\{c \in C : \exists e \in E : e \supseteq c\}|}{|C|}$$

$$R_{macro}(C, E) = P_{macro}(E, C)$$

6.3.2 Micro

Micro precision measures the purity of clusters in $C$, i.e., fraction of entities that appear in both canonicalization clusters and ground truth clusters based on the assumption that the majority of entities in a clusters are canonicalized correctly [7]. Micro recall is defined the same as micro precision except for the exchange of role of $C$ and $E$.

$$P_{micro}(C, E) = \frac{1}{N} \sum_{c \in C} \max_{e \in E} |c \cap e|$$

$$R_{micro}(C, E) = P_{micro}(E, C)$$

6.3.3 Pairwise

Pairwise precision is measured as the ratio of the number of hits in $C$ to the total possible pairs in $C$. A pair of entities produces a hit if they belong to the same cluster in the ground truth. Pairwise recall is defined the same as pairwise precision except for the exchange of role of $C$ and $E$.

$$P_{pair}(C, E) = \frac{\sum_{c \subseteq C} |\{(v, v') \in e, \exists e \in E, \forall (v, v') \in c\}|}{\sum_{c \subseteq C} |c|^2}$$

$$R_{pair}(C, E) = P_{pair}(E, C)$$

In general, precision and recall has a trade-off relation. A higher precision score usually
indicate the result clusters are more divisive and have lower recall score. A good canonicalization result should achieve a balance between two scores. Therefore, F1 score is also included in the evaluation.

6.3.4 Canonicalization Evaluation Example

Figure 5 is an illustrative example for evaluation metrics where c1, c2 and c3 denote the canonicalization clusters while e1, e2 and e3 denote the ground truth clusters.

![Illustrative Example for Evaluation Metrics](image)

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Macro</td>
<td>(\frac{2}{3})</td>
<td>(\frac{2}{3})</td>
<td>66.6</td>
</tr>
<tr>
<td>Micro</td>
<td>(\frac{5}{7})</td>
<td>(\frac{5}{7})</td>
<td>85.7</td>
</tr>
<tr>
<td>Pairwise</td>
<td>(\frac{4}{6})</td>
<td>(\frac{4}{7})</td>
<td>61.5</td>
</tr>
</tbody>
</table>

Table 3: Metrics Score for the Illustrative Example

For macro precision, only c2 and c3 are true subset of e2 and e3 respectively. c1 contains an entity New York City that is not in e1. Therefore, \(P_{macro} = \frac{2}{3}\) and \(R_{macro} = \frac{2}{3}\).
As for micro precision, the score is calculated on America, New York and California which are the majorities in \(c_1\), \(c_2\) and \(c_3\) respectively. \(c_1\) contains two America; \(c_2\) contains three New York; and \(c_3\) contains one California. Hence, \(P_{\text{micro}} = \frac{6}{7}\) and \(R_{\text{micro}} = P_{\text{micro}} = \frac{6}{7}\). The final result is shown in table 2.

For pairwise recall, \(e_1\) contains only one entity pair which is also a hit. \(e_2\) contains \(C_4^2 = 6\) among which \(C_3^2 = 3\) are hits. \(e_3\) does not contain any entity pair. So there are 7 pairs and 4 hits in total and \(R_{\text{pair}} = \frac{4}{7}\). Similarly, \(P_{\text{pair}} = \frac{4}{6}\).

### 6.4 Methods Compared

For open knowledge base canonicalization, MLCE is compared against the following methods.

- **GloVe**: Since the result of MLCE relies heavily on the interim result of GloVe. We decide to conduct an experiment that only use GloVe word embedding for canonicalization. We want to know want kind of improvement and how much improvement we can get from sentence level canonicalization. The set up of this method basically follows the design of word level canonicalization.

- **GHMS**: GHMS method is a classic canonicalization method that uses string comparison. It still has state of the art performance in dealing with more divisive knowledge bases. MLCE will degenerate to GHMS if the threshold of HAC algorithm is set to a very small value.

- **CESI**: CESI is one of the first methods that utilizes the embedding method. It has achieved state of the art performance on more clustered knowledge bases. Based on our understanding, CESI is an extended version of GloVe that embeds not only the word but also some side information such as PPDB and WordNet. We expect the performance of CESI to have the same pattern compared with GloVe.
7 Results

This section first presents the canonicalization result for the 5 datasets introduced in section 6.1 as well as their analysis. It then discusses two special cases where MLCE handles successfully. Finally, this section talks about difficulties remained in this project and proposes possible future work.

7.1 Results and Analysis

<table>
<thead>
<tr>
<th></th>
<th>Macro</th>
<th></th>
<th></th>
<th>Micro</th>
<th></th>
<th></th>
<th>Pairwise</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Precision</td>
<td>Recall</td>
<td>F1</td>
<td>Precision</td>
<td>Recall</td>
<td>F1</td>
<td>Precision</td>
<td>Recall</td>
<td>F1</td>
</tr>
<tr>
<td>Glove</td>
<td>0.46</td>
<td>0.84</td>
<td>0.60</td>
<td>0.82</td>
<td>0.97</td>
<td>0.89</td>
<td>0.86</td>
<td>0.96</td>
<td>0.90</td>
</tr>
<tr>
<td>GHMS</td>
<td>0.55</td>
<td>0.67</td>
<td>0.60</td>
<td>0.87</td>
<td>0.93</td>
<td>0.90</td>
<td>0.90</td>
<td>0.88</td>
<td>0.89</td>
</tr>
<tr>
<td>CESI</td>
<td>0.56</td>
<td>0.83</td>
<td>0.67</td>
<td>0.90</td>
<td>0.91</td>
<td>0.90</td>
<td>0.93</td>
<td>0.75</td>
<td>0.83</td>
</tr>
<tr>
<td>MLCE</td>
<td>0.66</td>
<td>0.57</td>
<td>0.61</td>
<td>0.90</td>
<td>0.91</td>
<td>0.90</td>
<td>0.94</td>
<td>0.94</td>
<td>0.94</td>
</tr>
</tbody>
</table>

Table 4: Canonicalization Result for Ambiguous Dataset

Table 4 shows the canonicalization result for Ambiguous dataset. As discussed in Section 6.1, Ambiguous dataset is a naive dataset that only contains about 400 golden entities and each with an average assertion number of 70. Because of the relatively large golden clusters, it is hard for canonicalization result to have pure entities. Consider the case where a result cluster of size 50 has only mistakenly classify 1 entity. This cluster will still be evaluated as an impure cluster. Therefore none of the four methods achieve a very satisfying macro score (see Table 4).

For micro score, all four methods achieve an F1 score around 0.90 because Ambiguous dataset does not contain many confusing ambiguous entities and it relatively easier to get the majority correct. As for pairwise score, the MLCE method achieves the best result. Actually, MLCE method achieves the best pairwise score among all methods for all datasets. Further explanation will be presented at the end of this section.
Table 5: Canonicalization Result for Reverb Freebase Dataset

<table>
<thead>
<tr>
<th>Method</th>
<th>Macro Precision</th>
<th>Macro Recall</th>
<th>Macro F1</th>
<th>Micro Precision</th>
<th>Micro Recall</th>
<th>Micro F1</th>
<th>Pairwise Precision</th>
<th>Pairwise Recall</th>
<th>Pairwise F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Glove</td>
<td>0.74</td>
<td>0.51</td>
<td>0.60</td>
<td>0.81</td>
<td>0.85</td>
<td>0.83</td>
<td>0.73</td>
<td>0.87</td>
<td>0.79</td>
</tr>
<tr>
<td>GHMS</td>
<td>0.88</td>
<td>0.35</td>
<td>0.50</td>
<td>0.89</td>
<td>0.80</td>
<td>0.85</td>
<td>0.84</td>
<td>0.77</td>
<td>0.80</td>
</tr>
<tr>
<td>CESI</td>
<td>0.77</td>
<td>0.52</td>
<td>0.62</td>
<td>0.83</td>
<td>0.86</td>
<td>0.84</td>
<td>0.77</td>
<td>0.86</td>
<td>0.81</td>
</tr>
<tr>
<td>MLCE</td>
<td>0.85</td>
<td>0.36</td>
<td>0.51</td>
<td>0.88</td>
<td>0.80</td>
<td>0.84</td>
<td>0.84</td>
<td>0.83</td>
<td>0.83</td>
</tr>
</tbody>
</table>

Table 6: Canonicalization Result for Reverb Wiki Dataset

<table>
<thead>
<tr>
<th>Method</th>
<th>Macro Precision</th>
<th>Macro Recall</th>
<th>Macro F1</th>
<th>Micro Precision</th>
<th>Micro Recall</th>
<th>Micro F1</th>
<th>Pairwise Precision</th>
<th>Pairwise Recall</th>
<th>Pairwise F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Glove</td>
<td>0.64</td>
<td>0.89</td>
<td>0.74</td>
<td>0.82</td>
<td>0.94</td>
<td>0.88</td>
<td>0.82</td>
<td>0.93</td>
<td>0.87</td>
</tr>
<tr>
<td>GHMS</td>
<td>0.88</td>
<td>0.60</td>
<td>0.72</td>
<td>0.91</td>
<td>0.85</td>
<td>0.88</td>
<td>0.88</td>
<td>0.78</td>
<td>0.83</td>
</tr>
<tr>
<td>CESI</td>
<td>0.66</td>
<td>0.90</td>
<td>0.76</td>
<td>0.84</td>
<td>0.95</td>
<td>0.89</td>
<td>0.84</td>
<td>0.91</td>
<td>0.88</td>
</tr>
<tr>
<td>MLCE</td>
<td>0.81</td>
<td>0.84</td>
<td>0.82</td>
<td>0.90</td>
<td>0.92</td>
<td>0.91</td>
<td>0.91</td>
<td>0.87</td>
<td>0.89</td>
</tr>
</tbody>
</table>

Table 7: Canonicalization Result for Reverb Intersect Dataset

<table>
<thead>
<tr>
<th>Method</th>
<th>Macro Precision</th>
<th>Macro Recall</th>
<th>Macro F1</th>
<th>Micro Precision</th>
<th>Micro Recall</th>
<th>Micro F1</th>
<th>Pairwise Precision</th>
<th>Pairwise Recall</th>
<th>Pairwise F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Glove</td>
<td>0.49</td>
<td>0.91</td>
<td>0.64</td>
<td>0.68</td>
<td>0.96</td>
<td>0.80</td>
<td>0.58</td>
<td>0.95</td>
<td>0.72</td>
</tr>
<tr>
<td>GHMS</td>
<td>0.78</td>
<td>0.85</td>
<td>0.82</td>
<td>0.87</td>
<td>0.92</td>
<td>0.89</td>
<td>0.82</td>
<td>0.84</td>
<td>0.84</td>
</tr>
<tr>
<td>CESI</td>
<td>0.65</td>
<td>0.91</td>
<td>0.76</td>
<td>0.80</td>
<td>0.95</td>
<td>0.87</td>
<td>0.77</td>
<td>0.92</td>
<td>0.84</td>
</tr>
<tr>
<td>MLCE</td>
<td>0.76</td>
<td>0.85</td>
<td>0.82</td>
<td>0.84</td>
<td>0.93</td>
<td>0.88</td>
<td>0.81</td>
<td>0.91</td>
<td>0.86</td>
</tr>
</tbody>
</table>

Table 5 and Table 6 are canonicalization results for Reverb Freebase dataset and Reverb Wiki dataset respectively. Although it is hard to argue which one a better dataset in terms of open KB canonicalization, the result indicates that using Wiki Link as ground truth generates more pure golden clusters since all four methods achieve a significantly better macro F1 score in Reverb Wiki dataset. Therefore, Reverb Wiki dataset is a better choice than Reverb Freebase dataset in terms of macro evaluation. As for micro score and pairwise score, all four methods have a consistent performance and our MLCE method have the best performance.

Table 7 shows the canonicalization result for Reverb Intersect Dataset. The nature of Reverb Intersect Dataset is a more divisive version of Reverb45k. Therefore, GHMS
method performs better than others since GHMS method tends to cluster entities with
the same meaning rather than entities with similar meaning, and thus create more dis-
itive clusters. Our MLCE method also achieves a satisfying result on this dataset except
for the micro score being slightly lower than GHMS.

Another thing worth discussion is the quality of Reverb Intersect Dataset. Although it
is a more divisive version Reverb45k, we do not see any improvement in macro eval-
uation. This is caused by the way we constructed Reverb Intersect Dataset. Consider
three entities $e_a$, $e_b$ and $e_c$, if $e_b$ only has Freebase ID which is the same as Freebase ID
of $e_a$ and $e_c$ only has Wiki Link which is the same of Wiki Link of $e_a$, then $e_a$, $e_b$ and $e_c$
will be in the same golden cluster. However, the relation between $e_b$ and $e_c$ cannot be
determined based on their own information. As a result, Reverb Intersect dataset may
create impure clusters during construction. And this problem become more severe in
Reverb Union dataset.

<table>
<thead>
<tr>
<th></th>
<th>Macro</th>
<th></th>
<th>Micro</th>
<th></th>
<th>Pairwise</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Precision</td>
<td>Recall</td>
<td>F1</td>
<td>Precision</td>
<td>Recall</td>
</tr>
<tr>
<td>Glove</td>
<td>0.68</td>
<td>0.75</td>
<td>0.72</td>
<td>0.80</td>
<td>0.77</td>
</tr>
<tr>
<td>GHMS</td>
<td>0.94</td>
<td>0.62</td>
<td>0.75</td>
<td>0.97</td>
<td>0.71</td>
</tr>
<tr>
<td>CESI</td>
<td>0.85</td>
<td>0.74</td>
<td>0.79</td>
<td>0.91</td>
<td>0.75</td>
</tr>
<tr>
<td>MLCE</td>
<td>0.90</td>
<td>0.60</td>
<td>0.72</td>
<td>0.93</td>
<td>0.71</td>
</tr>
</tbody>
</table>

Table 8: Canonicalization Result for Reverb Union Dataset

Table 8 shows the canonicalization result for Reverb Union Dataset. We notice a signif-
icant drop of three evaluation scores for all four methods. Consider the case mentioned
above, given three entities $e_a$, $e_b$ and $e_c$, $e_a$ has Freebase ID $f_1$ and Wiki Link $w_1$; $e_b$ has
Freebase ID $f_1$ and Wiki Link $w_2$; and $e_c$ has Freebase ID $f_2$ and Wiki Link $w_1$, we can
almost tell for sure $e_b$ and $e_c$ are different entities. However, these entities will end up
in the same golden clusters. Therefore, we argue that the evaluation of Reverb Union
dataset is inconclusive.

25
7.2 Case Study

This section studies two special cases where MLCE outperforms other state of the art method.

7.2.1 Mumbai and Bombay Case

Recall that Mumbai and Bombay case refers to two entities have the same meaning but different representations. Such case cannot be solved with GHMS since it only considers string similarity (see Figure 6&7). By utilizing the word embedding method, CESI can successfully handle this case (see Figure 14).

However, CESI solves this problem at a cost. Not only Mumbai and Bombay are put into the same cluster, but also other similar entities are put into the same cluster, such as Hong Kong and Macau (see Figure 9), Taipei and Kaoshiung (see Figure 10). Our MLCE method can put Mumbai and Bombay into the same cluster through word level canonicalization (see Figure 15) and successfully distinguish Hong Kong and Macau.
Taipei and Kaoshiung (see Figure 17).

Figure 15: Mumbai Cluster with MLCE

Figure 16: Hong Kong and Macau Clusters with MLCE

Figure 17: Taipei and Kaoshiung Cluster with MLCE
7.2.2 Airport Case

The Airport case refers to entities with different meaning but similar representations. Such problem can be hard to solve for CESI since it has no way to further distinguish similar entities (see Figure 18). By using sentence level canonicalization, our MLCE method generate decent clusters for different airports.

```
<table>
<thead>
<tr>
<th>id</th>
<th>sub:</th>
<th>src:</th>
</tr>
</thead>
<tbody>
<tr>
<td>41572</td>
<td>glasgow airport,</td>
<td>src: glasgow airport be in paisley</td>
</tr>
<tr>
<td>41718</td>
<td>glasgow airport,</td>
<td>src: glasgow airport be in glasgow</td>
</tr>
<tr>
<td>77709</td>
<td>new chitose airport,</td>
<td>src: new chitose airport be locate in hokkaido</td>
</tr>
<tr>
<td>147882</td>
<td>brussels airport,</td>
<td>src: brussels airport be the largest airport in belgium</td>
</tr>
<tr>
<td>36872</td>
<td>luton airport,</td>
<td>src: luton airport be north of london</td>
</tr>
<tr>
<td>31446</td>
<td>dubai airport,</td>
<td>src: dubai airport be seven kilometres from hotel</td>
</tr>
<tr>
<td>66652</td>
<td>manchester airport,</td>
<td>src: manchester airport be locate in manchester</td>
</tr>
<tr>
<td>85692</td>
<td>phuket international airport,</td>
<td>src: phuket international airport be the second busiest airport in thailand</td>
</tr>
<tr>
<td>85693</td>
<td>phuket international airport,</td>
<td>src: phuket international airport be locate in phuket</td>
</tr>
<tr>
<td>75459</td>
<td>mumbai international airport,</td>
<td>src: mumbai international airport be also know as chhatrapati shivaji intern</td>
</tr>
<tr>
<td>85691</td>
<td>phuket international airport,</td>
<td>src: phuket international airport be in phuket</td>
</tr>
<tr>
<td>20722</td>
<td>chicago hare international airport,</td>
<td>src: chicago hare international airport be locate in chicago</td>
</tr>
<tr>
<td>31449</td>
<td>dubai international airport,</td>
<td>src: dubai international airport be the hub for delta airlines</td>
</tr>
<tr>
<td>3379</td>
<td>airport,</td>
<td>src: airport be easily accessible from baltimore city</td>
</tr>
<tr>
<td>112653</td>
<td>washington dulles international airport,</td>
<td>src: washington dulles international airport be locate in dulle</td>
</tr>
<tr>
<td>3254</td>
<td>faro airport,</td>
<td>src: faro airport be locate in faro</td>
</tr>
<tr>
<td>45803</td>
<td>hartsfield jackson atlanta international airport,</td>
<td>src: hartsfield jackson atlanta international airport be</td>
</tr>
<tr>
<td>82837</td>
<td>paphos airport,</td>
<td>src: paphos airport serve limassol</td>
</tr>
<tr>
<td>16784</td>
<td>brussels international airport,</td>
<td>src: brussels international airport be at zaventem</td>
</tr>
<tr>
<td>4219</td>
<td>albuquerque international airport,</td>
<td>src: albuquerque international airport be locate in albuquerque</td>
</tr>
</tbody>
</table>
```

Figure 18: Airport Clusters with CESI

```
<table>
<thead>
<tr>
<th>id</th>
<th>sub:</th>
<th>src:</th>
</tr>
</thead>
<tbody>
<tr>
<td>35254</td>
<td>faro airport,</td>
<td>src: faro airport be locate in faro</td>
</tr>
<tr>
<td>35255</td>
<td>faro international airport,</td>
<td>src: faro international airport be the main gateway to algarve</td>
</tr>
<tr>
<td>82418</td>
<td>paphos international airport,</td>
<td>src: paphos international airport be locate in paphos</td>
</tr>
<tr>
<td>82417</td>
<td>paphos airport,</td>
<td>src: paphos airport serve limassol</td>
</tr>
<tr>
<td>3559</td>
<td>larnaca international airport,</td>
<td>src: larnaca international airport be closest to agora napa</td>
</tr>
<tr>
<td>60602</td>
<td>larnaca international airport,</td>
<td>src: larnaca international airport be closest to agora napa</td>
</tr>
<tr>
<td>3368</td>
<td>airport,</td>
<td>src: hare airport be own by chicago</td>
</tr>
<tr>
<td>20722</td>
<td>chicago hare international airport,</td>
<td>src: chicago hare international airport be locate in chicago</td>
</tr>
<tr>
<td>79813</td>
<td>hare airport,</td>
<td>src: hare airport be own by chicago</td>
</tr>
</tbody>
</table>
```

Figure 19: Airport Clusters with MLCE

7.3 Difficulties and Future Work

This section discusses two difficulties MLCE encountered and has not found a practical solution. We also propose some ideas for future work.

7.3.1 Blocking Criteria

One of the difficulties encountered in this project is the choice of blocking criteria. Recall the MLCE blocks entities by their subject. However, this design has a major drawback
when dealing with names. Again consider the case of Barack Obama, Michelle Obama and Obama, MLCE will first block them to three temporary canopies and then draw clusters. As a result, Obama will either be put with Michelle Obama or Barack Obama. However, we know in the real case, Obama can refer to either of them.

A possible solution to this problem is the hierarchy design. In the hierarchy blocking design, we first make a copy of Obama. Then we put one copy with Barack Obama and the other copy with Michelle Obama such that Obama get the chance to be classified with either of them. Hierarchy design could be quite complex if we consider more and more complicated cases. Another challenge is that all copies created should be merged in the final result/

7.3.2 Apple Problem

Apple problem refers to the case where the same representation refers to different entities such as the company Apple and fruit Apple. Existing methods all based on the assumption that same representation indicates same entity. To handle this problem, we propose fine-tuning of Bert as future work.

Figure 20: Fine-tuning Designs

Figure 20 shows two possible fine-tuning design. The left one is sentence level design
inspired by MRPC task [11]. Given a sentence pair, this design will output a class label for the sentence pair with 0 indicates subject in-equivalence and 1 indicates subject equivalence. The right one is the token level classification which is inspired by the NER task [12]. The only difference from sentence level classification is that this design will output the label for every token in the sentence pair.

8 Conclusion

Canonicalization is a relatively new research area but of much interest. Many state of the art canonicalization systems suffers from either Mumbai Bombay problem such as GHMS or Airport problem such as CESI. In this report, we present a new canonicalization system called MLCE which successfully handle both problems without hurting the accuracy.

We have conducted multiple experiments on four canonicalization methods GloVe, GHMS, CESI and MLCE on five different datasets named Ambiguous, Reverb Free-base, Reverb Wiki, Reverb Intersect and Reverb Union. Except for the Reverb Union dataset which turns out to be of low quality, our MLCE method has achieved state of the art performance on the rest four datasets. We also conduct two case studies on Mumbai Bombay problem and Airport problem. MLCE also yields a satisfying result regarding these two cases.

An important avenue for future work is to fine-tuning the Bert model to solve the Apple problem (same representation but different meanings). Existing canonicalization systems all base on the assumption that same representation implies same meaning. Thus, none of the current systems can deal with this problem. Two possible fine-tuning designs are introduced in section 7.3.2. These novel methods proposed, hopefully, can generate satisfactory results in the future.
9 Contribution

Many of the ideas and work are done in discussion and hard to split. Therefore, all group members agree on equal contribution to this project.
References


