High-Throughput and Language-Agnostic Entity Disambiguation and Linking on User Generated Data

Preeti Bhargava
Lithium Technologies | Klout
San Francisco, CA
preeti.bhargava@lithium.com

Nemanja Spasojevic
Lithium Technologies | Klout
San Francisco, CA
nemanja.spasojevic@lithium.com

Guoning Hu
Lithium Technologies | Klout
San Francisco, CA
guoning.hu@lithium.com

ABSTRACT

The Entity Disambiguation and Linking (EDL) task matches *entity mentions* in text to a unique Knowledge Base (KB) identifier such as a Wikipedia or Freebase id. It plays a critical role in the construction of a high quality information network, and can be further leveraged for a variety of information retrieval and NLP tasks such as text categorization and document tagging. EDL is a complex and challenging problem due to ambiguity of the mentions and real world text being multi-lingual. Moreover, EDL systems need to have high throughput and should be lightweight in order to scale to large datasets and run on off-the-shelf machines. More importantly, these systems need to be able to extract and disambiguate dense annotations from the data in order to enable an Information Retrieval or Extraction task running on the data to be more efficient and accurate. In order to address all these challenges, we present the Lithium EDL system and algorithm - a high-throughput, lightweight, language-agnostic EDL system that extracts and correctly disambiguates 75% more entities than state-of-the-art EDL systems and is significantly faster than them.

KEYWORDS

Entity Disambiguation, Entity Linking, Entity Resolution, Text Mining

ACM Reference format:
Preeti Bhargava, Nemanja Spasojevic, and Guoning Hu. 2016. High-Throughput and Language-Agnostic Entity Disambiguation and Linking on User Generated Data. In *Proceedings of LDOW 2017, Perth, Australia, April 2017 (WWW2017 Workshop: Linked Data on the Web),* 10 pages.

1 INTRODUCTION

In Natural Language Processing (NLP), Entity Disambiguation and Linking (EDL) is the task of matching *entity mentions* in text to a unique Knowledge Base (KB) identifier such as a Wikipedia or a Freebase id. It differs from the conventional task of Named Entity Recognition, which is focused on identifying the occurrence of an entity and its type but not the specific unique entity that the mention refers to. EDL plays a critical role in the construction of a high quality information network such as the Web of Linked Data [9]. Moreover, when any new piece of information is extracted from text, it is necessary to know which real world entity this piece refers to. If the system makes an error here, it loses this piece of information and introduces noise.

EDL can be leveraged for a variety of information retrieval and NLP tasks such as text categorization and document tagging. For instance, any document which contains entities such as *Michael Jordan* and *NBA* can be tagged with categories *Sports* and *Basketball*. It can also play a significant role in recommender systems which can personalize content for users based on the entities they are interested in.

EDL is complex and challenging due to several reasons:

- Ambiguity - The same entity mention can refer to different real world entities in different contexts. A clear example of ambiguity is the mention *Michael Jordan* which can refer to the basketball player in certain context or the machine learning professor from Berkeley. To the discerning human eye, it may be easy to identify the correct entity, but any EDL system attempting to do so needs to rely on contextual information when faced with ambiguity.
- Multi-lingual content - The emergence of the web and social media poses an additional challenge to NLP practitioners because the user generated content on them is often multi-lingual. Hence, any EDL system processing real world data on the web, such as user generated content from social media and networks, should be able to support multiple languages in order to be practical and applicable. Unfortunately, this is a challenge that has not been given enough attention.
- High throughput and lightweight - State-of-the-art EDL systems should be able to work on large scale datasets, often involving millions of documents with several thousand of entities. Moreover, these systems need to have low resource consumption in order to scale to larger datasets in a finite amount of time. In addition, in order to be applicable and practical, they should be able to run on off-the-shelf commodity machines.
- Rich annotated information - All information retrieval and extraction tasks are more efficient and accurate if the underlying data is rich and dense. Hence, EDL systems need to ensure that they extract and annotate many more entities and of different types (such as professional titles, sports, activities etc.) in addition to just named entities (such as persons, organizations, locations etc.) However, most existing systems focus on extracting named entities only.

In this paper, we present our EDL system and algorithm, hereby referred to as the Lithium EDL system, which is a high-throughput, lightweight and language-agnostic EDL

WWW2017 Workshop: Linked Data on the Web, Perth, Australia

system that extracts and correctly disambiguates 75% more entities than state-of-the-art EDL systems and is significantly faster than them.

1.1 Related Work

EDL has been a well studied problem in literature and has gained a lot of attention in recent years. Approaches that disambiguate entity mentions with respect to Wikipedia date back to Bunescu and Pasca's work in [3]. Cucerzan [5] attempted to solve the same problem by using heuristic rules and Wikipedia disambiguation markups to derive mappings from display names of entities to their Wikipedia entries. However, this approach doesn't work when the entity is not well defined in their KB. Milne and Witten [11] refined Cucerzan's work by defining topical coherence using normalized Google Distance [4] and only using 'unambiguous entities' to calculate topical coherence.

Recent approaches have focused on exploiting statistical text features such as mention and entity counts, entity popularity and context similarity to disambiguate entities. Spotlight [6] used a maximum likelihood estimation approach using mention and entity counts. To combine different types of disambiguation knowledge together, Han and Sun [8] proposed a generative model to include evidences from entity popularity, mention-entity association and context similarity in a holistic way. More recently, systems like AIDA [16] and AIDA-light [12] have proposed graphical approaches that employ these statistical measures and attempt the disambiguation of multiple entries in a document simultaneously. Bradesco et al. [2] followed an approach similar to AIDA-light [12] but limited the entities of interest to people and companies. However, a major disadvantage of such approaches is that their combinatorial nature results in intractability, which makes them harder to scale to very large datasets in a finite amount of time. In addition, all these systems do not support multi-lingual content which is very common nowadays due to the prolificity of user generated content on the web.

Our work differs from the existing work in several ways. We discuss these in the contributions outlined below.

1.2 Contributions

Our contributions in this paper are:

- Our EDL algorithm uses several context-dependent and context-independent features, such as mention-entity cooccurrence, entity-entity cooccurrence, entity importance etc., to disambiguate mentions to their respective entities.
- In contrast to several existing systems such as Google Cloud NL API [1], OpenCalais [2] and AIDA [16], our EDL system recognizes several types of entities (professional titles, sports, activities etc.) in addition to named entities (people, places, organizations etc.). Our experiments (Section 7.2) demonstrate that it recognizes and correctly disambiguates about 75% more entities than state-of-the-art systems. Such richer and denser annotations are particularly useful in understanding the user generated content on social media to model user conversations and interests.
- Our EDL algorithm is language-agnostic and currently supports 6 different languages including English, Arabic, Spanish, French, German, and Japanese[3]. As a result, it is highly applicable to process real world text such as multi-lingual user generated content from social media. Moreover, it does not need any added customizations to support additional languages. In contrast, systems such as AIDA [16] and AIDA-light [12] need to be extended by additional components in order to support other languages such as Arabic [17].
- Our EDL system has high throughput and is very lightweight. It can be run on an off-the-shelf commodity machine and scales easily to large datasets. Experiments with a dataset of 910 million documents showed that our EDL system took about 2.2ms per document (with an average size of 169 bytes) on a 2.5 GHz Xeon processor (Section 6.3). Moreover, our experiments demonstrate that our system's runtime per unique entity extracted is about 3.5 times faster than state-of-the-art systems such as AIDA [16].

2 KNOWLEDGE BASE

Our KB consists of about 1 million Freebase[4] machine ids for entities. These were chosen from a subset of all Freebase entities that map to Wikipedia entities. We prefer to use Freebase rather than Wikipedia as our KB since in Freebase, the same id represents a unique entity across multiple languages. Due to limited resources and usefulness of the entities, our KB contains approximately 1 million most important entities from among all the Freebase entities. This gives us a good balance between coverage and relevance of entities for processing common social media text. Section 3.3.1 explains how entity importance is calculated, which enables us to rank the top 1 million Freebase entities.

In addition to the KB entities, we also employ two special entities: **NIL** and **MISC**. **NIL** entity indicates that there is no entity associated with the mention, eg. mention 'the' within the sentence may link to entity **NIL**. This entity is useful especially when it comes to dealing with stop words and false positives. **MISC** indicates that the mention links to an entity which is outside the selected entity set in our KB.

3 SYSTEM ARCHITECTURE

This paper is focused on describing the Lithium EDL system. However, the EDL system is a component of a larger Natural Language Processing (NLP) pipeline, hereby referred to as the Lithium NLP pipeline, which we describe briefly here. Figure 1 shows the high level overview of the Lithium NLP pipeline. It consists of several Text Preprocessing stages before EDL.

[1] https://cloud.google.com/natural-language/

[2] http://www.opencalais.com/

[3] Our EDL system can easily support more languages with the ready availability of ground truth data in them

[4] Freebase was a standard community generated KB until June 2015 when Google deprecated it in favor of the commercially available Knowledge Graph API.

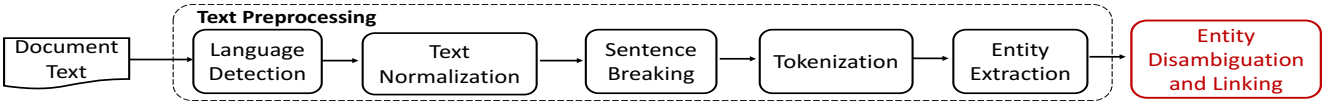


Figure 1: Overview of the Lithium NLP pipeline

3.1 Text Preprocessing

The Lithium NLP pipeline processes an input text document in the following stages before EDL:

- **Language Detection** - This stage detects the language of the input document using a naive Bayesian filter. It has a precision of 99% and is available on GitHub[5].
- **Text Normalization** - This stage normalizes the text by escaping unescaped characters and replacing some special characters based on the detected language. For example, it replaces non-ASCII punctuations with spaces and converts accents to regular characters for English.
- **Sentence Breaking** - This stage breaks the normalized text into sentences using the Java Text API[6]. This tool can distinguish sentence breakers from other marks, such as periods within numbers and abbreviations, according to the detected language.
- **Tokenization** - This stage converts each sentence into a sequence of tokens via the Lucene Standard Tokenizer[7].
- **Entity Extraction** - This stage captures mentions in each sentence that belong to precomputed offline dictionaries. Please see Section 3.3.1 for more details about dictionary generation. A mention may contain a single token or several consecutive tokens, but a token can belong to at most one mention. Often there are multiple ways to break a sentence into a set of mentions. To make this task computationally efficient, we apply a simple greedy strategy that analyzes windows of n-grams (n $\in$ [1,6]) and extracts the longest mention found in each window.

An extracted mention maps to multiple candidate entities. Our pipeline determines the best entity for each mention in the EDL phrase, which is described in Section 3.3.

3.2 Data Set Generation

Since our goal here is to build a language-agnostic EDL system, we needed a dataset that scales across several languages and also has good entity density and coverage. Unfortunately, such a dataset is not readily available. Hence, we generated a ground truth data set for our EDL system, the Densely Annotated Wikipedia Text (DAWT)[8] [13], using densely Wikified [10] or annotated Wikipedia articles. Wikification is entity linking with Wikipedia as the KB. We started with Wikipedia data dumps[9], which were further enriched by introducing more hyperlinks in the existing document structure. Our main goals when building this data set were to maintain high precision and increase linking coverage. As a last step, the hyperlinks to Wikipedia articles in a specific language were replaced with links to their Freebase ids to adapt to our KB. The densely annotated Wikipedia articles had on an average 4.8 times more links than the original articles.

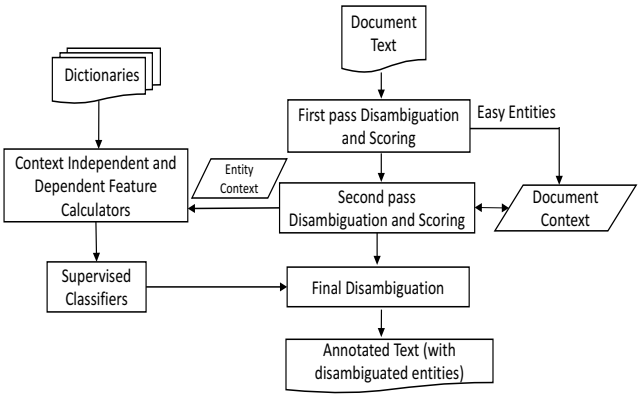


Figure 2: System architecture of the Entity Disambiguation and Linking stage

3.3 Entity Disambiguation and Linking

The system architecture of the EDL stage is shown in Figure 2. Similar to the approach employed by AIDA-light [12], it employs a two-pass algorithm (explained in detail in Section 4) which first identifies a set of *easy* mentions, which have low ambiguity and can be disambiguated and linked to their respective entities with high confidence. It then leverages these easy entities and several context dependent and independent features to disambiguate and link the remaining *hard* mentions. However, unlike AIDA-light [12], our approach does not use a graph based model to jointly disambiguate entities because such approaches can become intractable with increase in the size of the document and number of entities. In addition, our EDL problem is posed as a classification rather than a regression problem as in AIDA-light [12].

The EDL stage consists of the following components:

3.3.1 ***Offline Dictionaries Generation***. Our EDL system uses several dictionaries capturing language models, probabilities and relations across entities and topics. These are generated by offline processes leveraging various multi-lingual data sources to generate resource files. These are:

[5]https://github.com/shuyo/language-detection
[6]https://docs.oracle.com/javase/7/docs/api/java/text/BreakIterator.html
[7]http://lucene.apache.org/core/4_5_0/analyzers-common/org/apache/lucene/analysis/standard/StandardTokenizer.html
[8]DAWT and other derived datasets are available for download at: https://github.com/klout/opendata/tree/master/wiki_annotation.
[9]https://dumps.wikimedia.org/

- **Mention-Entity Cooccurrence** - This dictionary is derived using the DAWT data set [13]. Here, we estimate the prior probability that a mention M_i refers to an entity E_j (including **NIL** and **MISC**) with respect to our KB and corpora. It is equivalent to the cooccurrence probability of the mention and the entity:
$$\frac{count(M_i \rightarrow E_j)}{count(M_i)}$$
We generate a separate dictionary for each language. Moreover, since DAWT is 4.8 times denser than Wikipedia, these dictionaries capture several more mentions and are designed to be exhaustive across several domains.
- **Entity-Entity Cooccurrence** - This dictionary is also derived using DAWT. In this case, we capture co-occurrence frequencies among entities by counting all the entities that simultaneously appear within a sliding window of 50 tokens. Moreover, this data is accumulated across all languages and is language independent in order to capture better relations and create a smaller memory footprint when supporting additional languages. Also, for each entity, we consider only the top 30 co-occurring entities which have at least 10 or more co-occurrences across all supported languages.
- **Entity Importance** - The entity importance score [1] is derived as a global score identifying how important an extracted entity is for a casual observer. This score is calculated using linear regression with features capturing popularity within Wikipedia links, and importance of the entity within Freebase. We used signals such as Wiki page rank, Wiki and Freebase incoming and outgoing links, and type descriptors within knowledge base etc.
- **Topic Parent** - The Klout Topic Ontology[10] is a manually curated ontology built to capture social media users' interests [15] and expertise scores [14] across multiple social networks. As of December 2016, it consists of roughly 7,500 topic nodes and 13,500 edges encoding hierarchical relationships among them. The Topic Parents dictionary contains the parent topics for each topic within this ontology.
- **Entity To Topic Mapping** - This dictionary essentially contains topics from the Klout Topic Ontology that are associated with the different entities in our KB. E.g. Michael Jordan, the basketball player, will be associated with the topics 'Football' and 'Sports'. We generate this dictionary via a weighted ensemble of several algorithms that employ entity co-occurrence and propagate the topic labels. A complete description of these algorithms is beyond the scope of this paper.

3.3.2 *Context*.

- **Document context** - As mentioned earlier, the Lithium EDL system relies on disambiguating a set of *easy* mentions in the document which are then leveraged to disambiguate the *hard* mentions. Thus, for each document, we maintain a *document context* $C(T_i)$ which includes all the easy entities in the document text that have been disambiguated. This context also includes cached pairwise feature scores for the context dependent features between the easy and hard entities (see Section 4.2.1 for a description of the context dependent features).
- **Entity context** - For each candidate entity E_k of a hard mention, we define an *entity context* $C'(E_k)$ which includes the position of the corresponding mention in the document, the index number of the candidate entity as well as an *easy entity window* $\mathbb{E}_k$ surrounding the hard mention. The appropriate window size W is determined by parameter tuning on a validation set.

3.3.3 *Supervised Classifiers*.

We pose our EDL problem as a binary classification problem for the following reason: For each mention, only one of the candidate entities is the correct label entity. Our ground truth data set provides the labeled correct entity but does not have any scores or ranked order for the candidate entities. Hence, we pose this problem as predicting one of the two labels {*True, False*} for each candidate entity (where *True* indicates it is the correctly disambiguated entity for a mention and *False* indicates that it is not).

Using the process described in Section 3.2, we generated a ground truth training set of 70 English Wikipedia pages which had a total of 43,662 mentions and 147,236 candidate entities. We experimented with several classifiers such as Decision Trees, Random Forest, k-Nearest Neighbors and Logistic Regression on this training set. Decision Trees and Logistic Regression outperformed most of the classifiers. While Random Forest was as accurate as the Decision Tree classifier, it was computationally more expensive. Hence, we use Decision Tree and Logistic Regression in the Lithium EDL system.

4 ENTITY DISAMBIGUATION AND LINKING ALGORITHM

Algorithm 1 describes the Lithium EDL two-pass algorithm. We discuss it in detail now (the design choices for various parameters are explained in Section 5).

4.1 First pass

The first pass of the algorithm iterates over all mentions in the document text and disambiguates mentions that have:

- Only one candidate entity: In this case, the algorithm disambiguates the mention to the lone candidate entity.
- Two candidate entities with one being **NIL**/**MISC**: In this case, the algorithm disambiguates the mention to the candidate entity with high *Mention-Entity-Cooccurr* prior probability (above λ_1 - Easy Mention Disambiguation threshold with **NIL**).
- Three or more candidate entities with one entity mapping with very high prior: In this case, the algorithm disambiguates the mention to the candidate

[10] https://github.com/klout/opendata

Algorithm 1: Lithium EDL algorithm

Input: Text T_i with extracted mentions $\mathbb{M}_{all}$ and a set of candidate entities for each mention
Output: Text T_i with extracted mentions $\mathbb{M}_{all}$ and a unique disambiguated entity for each mention

```
// First pass - Disambiguate the easy mentions
```

$\mathbb{M}_{easy} \leftarrow$ Easy mentions obtained from the first pass on T_i;
$\mathbb{E}_{easy} \leftarrow$ Disambiguated easy entities obtained from the first pass on T_i;
Document Context $C(T_i) \leftarrow C(T_i) + \mathbb{E}_{easy}$;
$\mathbb{M}_{hard} \leftarrow \mathbb{M}_{all}$ - $\mathbb{M}_{easy}$;

```
// Second pass - Iterate over the hard mentions
```

foreach *Mention* $M_j \in \mathbb{M}_{hard}$ **do**
  $\mathbb{H}_j \leftarrow$ Candidate entities of M_j;

```
  // Iterate over the candidate entities of a hard mention
```

  foreach *Entity* $E_k \in \mathbb{H}_j$ **do**
    Entity Context C'(E_k) $\leftarrow$ C'(E_k) + $\mathbb{E}'_k$ (set of easy entities in a window around E_k) ;
    $F_{E_k} \leftarrow$ Generate feature vector of context independent and dependent features values for E_k using C'(E_k);
    Classify F_{E_k} as one of {*True, False*} using Decision Tree classifier;
    $S_{E_k} \leftarrow$ Final score for E_k generated using Logistic Regression model weights;
    Add S_{E_k} to set S_j (Set of candidate entity scores for M_j);
  end

```
  // Final disambiguation - select one of the candidate entities as disambiguated entity D_j for M_j
```

  if *Only one* $E_k \in \mathbb{H}_j$ *labeled as* True **then**
    $D_j \leftarrow E_k$ labeled as *True*;
  else
    if *Multiple* E_k *labeled as True* **then**
      $D_j \leftarrow$ Highest scoring E_k labeled as *True*;
    else if *None of* E_k *labeled as True* **then**
      $D_j \leftarrow \arg\max\ (S_{E_k})$;
    if D_j *is* ***NIL*** *and* NIL_MARGIN_GAIN $<$ *threshold* **then**
      $D_j \leftarrow \arg\max\ (S_j$ - $S_{D_j})$;
end
return *Text with extracted mentions and disambiguated entities;*

entity with high *Mention-Entity-Cooccurr* prior probability (above λ_2 - Easy Mention Disambiguation threshold).

Mentions disambiguated in the first pass constitute the set $\mathbb{M}_{easy}$ and their corresponding disambiguated entities constitute the set $\mathbb{E}_{easy}$. The remaining ambiguous mentions constitute the set $\mathbb{M}_{hard}$ and are disambiguated in the second pass.

4.2 Second pass

The second pass of the algorithm uses several context-independent and context-dependent features as well as supervised classifiers to label and score the candidate entities for each hard mention and finally disambiguate it.

4.2.1 ***Features.*** We use several language agnostic features to classify each candidate entity for each hard mention as 'True' or 'False'. These include both context-independent (useful for disambiguating and linking entities in short and sparse texts such as tweets) as well as context-dependent features (useful for disambiguating and linking entities in long and rich text). Each feature produces a real value in [0.0,1.0].

The context independent features are:

- **Mention-Entity Cooccurrence** (*Mention-Entity-Cooccurr*) - This feature value is equal to the *Mention-Entity-Cooccurr* prior probability.
- **Mention-Entity Jaccard Similarity** (*Mention-Entity-Jaccard*) - This reflects the similarity between the mention M_i and the representative name of a candidate entity E_j. The mention and the entity display names are first tokenized and the Jaccard similarity is then computed between the token sets as

$$\frac{Tokens(M_i) \cap Tokens(E_j)}{Tokens(M_i) \cup Tokens(E_j)}$$

  For instance, the mention *Marvel* could refer to the entities *Marvel Comics* or *Marvel Entertainment*, both of which have a Jaccard Similarity of 0.5 with the mention.
- **Entity Importance** (*Entity-Importance*) - This reflects the importance or the relevance of the candidate entity as determined by an entity scoring and ranking algorithm [1] which ranks the top 1 million entities occurring in our KB. For instance, the entity *Apple Inc.* has an importance of 0.66 while *Apple (fruit)* has an importance of 0.64 as ranked by the Entity Scoring algorithm.

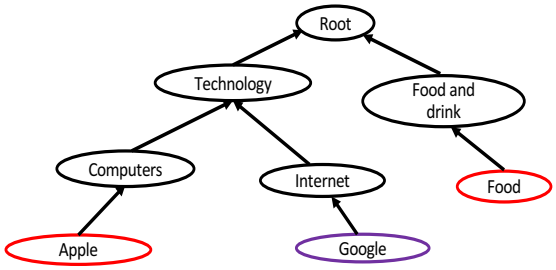


Figure 3: Semantic distance between topics in Klout Topic Ontology Space

For the following context dependent features, we assume that for a candidate entity E_i, we maintain an entity context C'(E_i) which contains a window $\mathbb{E}'_i$ of W disambiguated easy entities immediately surrounding E_i.

- **Entity Entity Cooccurrence** (*Entity-Entity-Cooccurr*) - This feature value is equal to the averaged co-occurrence of a candidate entity with the disambiguated easy entities in $\mathbb{E}'_i$ and is computed as:

$$\frac{\sum_{j=1}^{W} Co-occurrence-count(E_i, E_j)}{W} \forall E_j \in \mathbb{E}'_i$$

- **Entity Entity Topic Semantic Similarity** (*Entity-Entity-Topic-Sim*) - As mentioned in Section 3.3.1, each entity in our KB is associated with a finite number of topics in our topic ontology. For instance, entity *Apple Inc.* maps to the topic 'Apple' and *Google Inc.* maps to the topic 'Google' while '*Apple (fruit)*' will map to the topic 'Food'. Figure 3 shows a partial view of the ontology for the above mentioned topics.

  For each candidate entity E_i of a hard mention M_i, we compute the minimum *semantic distance* of its topics with topics of each entity in $\mathbb{E}'_i$ over all possible paths in our topic ontology space. The similarity is the inverse of the distance. For instance, consider the hard mention *Apple*, having two candidate entities - *Apple Inc.* and *Apple (fruit)* for it, and $\mathbb{E}'_i$ containing the entity *Google Inc.* which has been disambiguated. As shown in Figure 3, the semantic distance between the topics for *Apple Inc.* and *Google Inc.* is 4 while the semantic distance between the topics for *Apple (fruit)* and *Google Inc.* is 5. As a result, it is more likely that *Apple* disambiguates to *Apple Inc.*

  Thus, we first determine the set of topics $\mathbb{T}_i$ that the candidate entity E_i is associated with. For each entity E_j in $\mathbb{E}'_i$, we generate the set of topics $\mathbb{T}_j$. The feature value is computed as

$$\max \frac{1}{distance(t_i, t_j)} \forall t_i \in \mathbb{T}_i, t_j \in \mathbb{T}_j$$

4.2.2 ***Classification and Scoring***. As a penultimate step in the second pass, the computed features are combined into a feature vector for a candidate entity and the Decision Tree classifier labels the feature vector as 'True' or 'False'. In addition, for each candidate entity, we also generate final scores using weights generated by the Logistic Regression classifier that we trained in Section 3.3.3. We use an ensemble of the two classifiers in the final disambiguation step as it helps overcome the individual bias of each classifier.

4.2.3 ***Final Disambiguation***. The final disambiguation step needs to select one of the labeled candidate entities as the disambiguated entity for the mention. However, multiple cases arise at the time of disambiguation:

- Only one candidate entity is labeled as 'True'- Here, the algorithm selects that entity as the disambiguated entity for the given mention.
- Multiple candidate entities labeled as 'True' - Here, the algorithm selects the highest scoring entity (from among those labeled 'True') as the disambiguated entity except when this entity is **NIL**/**MISC**. In that case, the algorithm checks the *margin of gain* or the score difference between the **NIL**/**MISC** entity and the next highest scoring entity that is labeled 'True'. If the margin of gain is less than a threshold (less than **NIL** margin of gain threshold, λ_3) then the next highest scoring entity (from among those labeled 'True') is selected.
- All candidate entities labeled as 'False' - Here, the algorithm selects the highest scoring entity as the disambiguated entity except when this entity is **NIL**/**MISC**. In that case, the algorithm checks the margin of gain for this entity over the next highest scoring entity. If the margin of gain is less than a threshold (less than **NIL** margin of gain threshold, λ_3) then the next highest scoring entity is selected.

4.3 Demonstrative Example

To demonstrate the efficacy of our algorithm, let's disambiguate the sample text: *"Google CEO Eric Schmidt said that the competition between Apple and Google and iOS vs. Android is 'the defining fight of the tech industry.' ".*

Figure 4 walks through the disambiguation of the sample text. The Text Preprocessing stages extract the mentions (highlighted in bold) and generate the candidate entities and the prior cooccurrence scores for each mention[11]. As shown, the extracted mentions and their candidate entities are:

- *Google* - **NIL** and *Google Inc.*
- *CEO* - **NIL** and *Chief Executive*
- *Eric Schmidt* - **NIL** and *Eric Schmidt*
- *Apple* - **NIL**, *Apple (fruit)*, *Apple Inc.* and *Apple Records*
- *iOS* - **NIL** and *iOS*
- *Android* - **NIL**, *Android (OS)* and *Android(robot)*
- *tech industry* - *Technology*

In the first pass, the algorithm disambiguates the easy mentions. Based on their high prior scores and number of candidate entities, it disambiguates *Eric Schmidt*, *iOS* and *tech industry* (highlighted in color) to their correct entities. In the second pass, it uses the easy mention window and computes several context dependent and independent features to score and classify the candidate entities of the hard mentions. Note that for the purpose of clarity and simplicity, we are not walking through the feature and final score computation.

[11]Though our algorithm utilizes the Freebase machine id for each candidate entity, we only show the entity name for clarity.

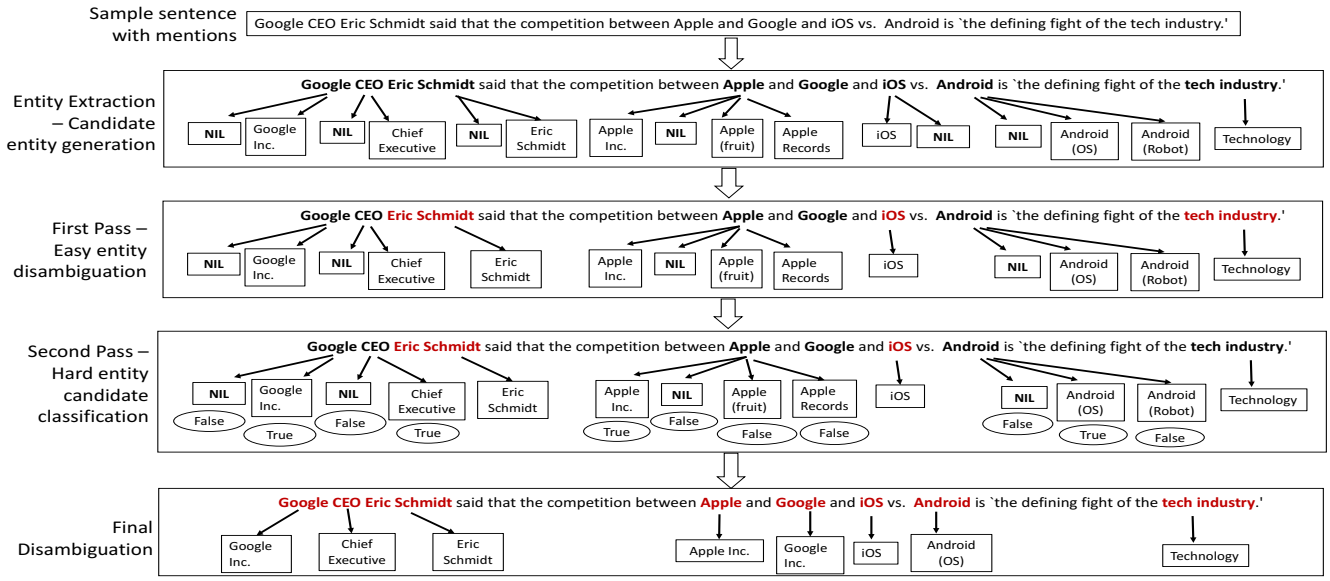


Figure 4: Disambiguation of a sample sentence (best viewed in color)

As shown, for the remaining hard entities, it has classified the candidate entities as 'True' or 'False'. In the final disambiguation step, it selects one of the labeled entities as the correct disambiguated entity. In the sample sentence, for all the mentions, only one of the candidate entities is labeled as 'True', and hence the algorithm selects that entity as the disambiguated entity for each mention.

5 PARAMETER TUNING

Our algorithm uses four different hyperparameters - 2 in the first pass and 2 in the second pass. These are:

- Easy mention disambiguation threshold with **NIL** (λ_1) - This threshold is used to disambiguate easy mentions which have 2 candidate entities and one of them is the **NIL** entity.
- Easy mention disambiguation threshold (λ_2) - This threshold is used to disambiguate easy mentions which have 3 or more candidate entities but the mention maps to one of them with a very high prior probability.
- **NIL** margin of gain threshold (λ_3) - This threshold is used in the second pass to disambiguate entities when multiple or none of the candidates are labeled 'True'.
- Window size (W) - This parameter represents the size of the easy entity window around each hard entity.

Using the process described in Section 3.2, we generated a ground truth validation set of 10 English Wikipedia pages which had a total of 7242 mentions and 23,961 candidate entities. We used parameter sweeping experiments to determine the optimal value of these parameters. We measured the performance (in terms of precision, recall and f-score) of the algorithm on the validation set with different parameter settings and picked the parameter values that had the best performance. Based on our experiments, we set the optimal value of λ_1 as 0.75, λ_2 as 0.9, W as 400 and λ_3 as 0.5.

	Ground Truth Label	
Predicted Label	Correct Entity	**NIL**
Correct Entity	TP	FP
Wrong Entity	FP	FP
NIL	FN	TN

Table 1: Confusion matrix for our EDL system

6 EVALUATION

6.1 Test data

Using the process described in Section 3.2, we generated a ground truth test set of 20 English Wikipedia pages which had a total of 18,773 mentions.

6.2 Metrics

We use standard performance metrics like precision, recall, f-score and accuracy to evaluate our EDL system on the test set. However, due to our problem setup, we calculate true positives, false positives, and true negatives and false negatives in an unconventional way as shown in Table 1. Precision, recall, f-score and accuracy are calculated in the standard format as: $\mathrm{P} = \frac{t_p}{t_p + f_p}$, $\mathrm{R} = \frac{t_p}{t_p + f_n}$, $\mathrm{F1} = \frac{2 \times \mathrm{P} \times \mathrm{R}}{\mathrm{P} + \mathrm{R}}$ and Accuracy = $\frac{t_p + t_n}{t_p + t_n + f_p + f_n}$

6.3 Results

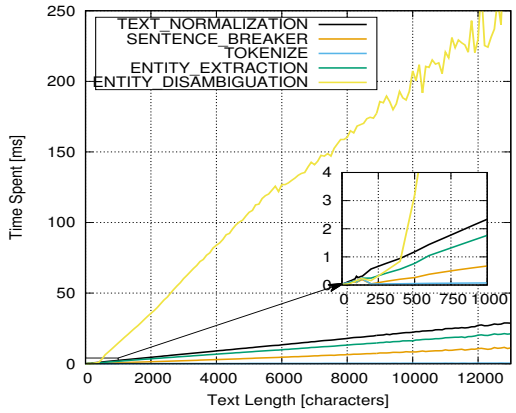
We compute the performance metrics for individual features as well as for various feature sets on our English language test set to assess their impact. Table 2 shows the feature effectiveness results for our algorithm. As evident from the results, *Mention-Entity-Cooccurr* has the biggest impact on the performance of the algorithm among all individual features as it has the highest individual precision and f-score.

Features	Precision	Recall	F-score	Accuracy
Mention-Entity-Cooccurr (context independent)	0.65	0.75	0.70	0.62
Mention-Entity-Jaccard (context independent)	0.47	0.93	0.48	0.63
Entity-Importance (context independent)	0.50	0.90	0.50	0.65
Entity-Entity-Cooccurr (context dependent)	0.54	0.91	0.54	0.68
Entity-Entity-Topic-Sim (context dependent)	0.49	0.88	0.49	0.63
Combined Context independent features	0.63	0.83	0.62	0.72
Combined Context dependent features	0.52	0.92	0.52	0.66
All features	0.63	0.87	0.73	0.64

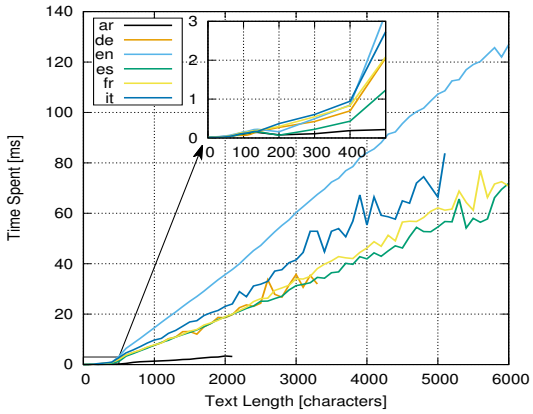
Table 2: Precision, recall, f-score and accuracy for different features and feature sets on our test set (English only)

Language	Precision	Recall	F-score	Accuracy
English	0.63	0.87	0.73	0.64
French	0.59	0.86	0.70	0.6
German	0.63	0.90	0.74	0.64
Spanish	0.58	0.89	0.70	0.60
Japanese	0.73	0.88	0.80	0.74

Table 3: Precision, recall, f-score and accuracy across various languages



(a) Processing Stages for English



(b) Different Languages for Entity Disambiguation

Figure 5: Processing times as function of text length

When combined, the context independent features combined have higher precision and f-score than the context dependent features. This could be due to the fact that in shorter text documents, there may not be enough easy mentions disambiguated in the first pass. Since the context dependent features rely on the easy entity window for computation, their performance will be impacted. However, when all these features are taken together, the overall performance improves even further. This demonstrates that context is an important factor in entity disambiguation and linking. Our final algorithm, which utilizes all the context dependent and independent feature sets, has a precision of 63%, recall of 87% and f-score of 73%.

Table 3 shows the performance of the Lithium EDL system across various languages. We note that the test datasets for these languages are smaller. However, the algorithm's performance is comparable to that for the English dataset.

6.4 Runtime Performance

The Lithium EDL system has been built to run in a bulk manner as well as a REST API service. The two major challenges that we faced while developing the system were the volume of new data that we process in bulk daily and limited computational capacity. These challenges had a significant influence on our system design and algorithmic approach.

As a demonstrative example, the most consuming task in our MapReduce cluster processes around 910 million documents, with an average document size of 169 bytes, taking about 2.2ms per document. Our MapReduce cluster has around 150 Nodes each having a 2.5 GHz Xeon processor. The processing is distributed across 400 reducers. The Reduce step takes about 2.5 hrs. Each reducer task runs as a single thread with an upper bound of 7GB on memory where the processing pipeline and models utilize 3.7GB.

A more detailed breakdown of the computational performance of our system as a function of document length is shown in Figure 5. The overall performance of the system is a linear function of text length. We also analyze this performance for different languages as well as for different stages of the Lithium NLP pipeline. We can see that the computation is slowest for English since it has the maximum number of entities [13].

7 COMPARISON WITH OTHER COMMERCIAL SYSTEMS

Currently, due to limited resources at our end and due to inherent differences in KB, data and text preprocessing stages, a direct comparison of the Lithium EDL system's performance (in terms of precision, recall and f-score) with other

	Lithium	Google NL	Both
English	5548	1501	1062
Spanish	2410	1152	839
Japanese	1631	801	549
All	9589	3454	2450

Table 4: Comparison of Lithium EDL and Google Cloud NL API

	Lithium	OpenCalais
English	5548	1295
Spanish	2410	885
French	3341	1161
All	11299	3341

Table 5: Comparison of Lithium EDL and OpenCalais API

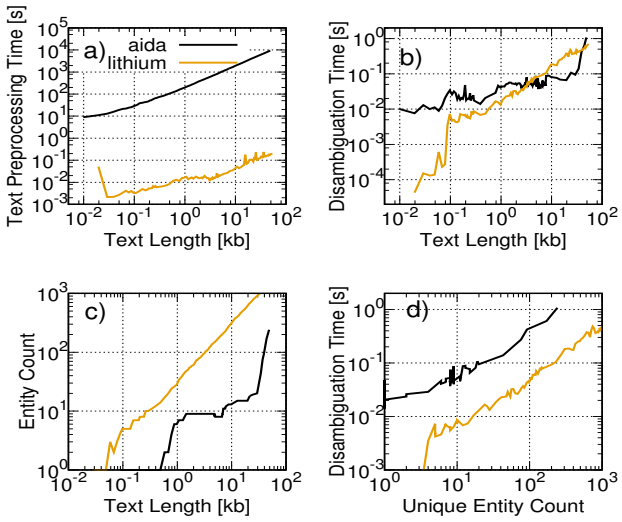


Figure 6: AIDA and Lithium NLP Pipeline Comparisons. a) Text preprocessing runtime; b) Disambiguation runtime; c) Extracted entity count; d) Disambiguation runtime as function of entity count;

commercial systems, such as Google Cloud NL API, OpenCalais and AIDA, is not possible. Hence, we compare our system with them on a different set of metrics.

7.1 Comparison on languages

While the Lithium EDL system supports about 6 different languages (English, Arabic, Spanish, French, German, Japanese), Google Cloud NL API supports mainly 3 languages: English, Spanish, and Japanese. Similarly, OpenCalais supports only English, Spanish, and French while AIDA only supports English and Arabic.

7.2 Comparison on linked entity density

A major advantage of our system is the ability to discover and disambiguate a much larger number of entities compared to other state-of-the-art systems. As a demonstration, we compared our result with Google Cloud NL API and OpenCalais[12]. In particular, we ran both APIs on documents in our test data set with the common subset of languages that they supported.

Table 4 compares the total number of unique entities disambiguated by Lithium EDL system and those by Google NL. An entity from Google NL is considered to be disambiguated if it was associated with a Wikipedia link. Column **Both** shows the numbers of entities that were disambiguated by both systems. Most entities disambiguated by Google NL were also disambiguated by our system. In addition, our system disambiguated several more entities. Based on the the precision of our system, we can estimate that at least 6080 disambiguated entities from our system are correct. This implies that Google NL missed more than 2600 entities that were correctly disambiguated by our system. Thus, our system correctly disambiguated at least 75% more entities than Google NL.

Table 5 shows a similar comparison between our system and OpenCalais. Every entity from OpenCalais API is considered to be disambiguated. However, since OpenCalais entity does not link the disambiguated entities to Wikipedia or Freebase but to their own proprietary KB, we cannot determine which entities were discovered by both the systems. Nevertheless, based on the precision of our system, at least 3500 entities that were correctly disambiguated by our system, were missed by OpenCalais, which is significantly more than the number of entities they detected.

7.3 Comparison on runtime

We compared the runtime performance of the Lithium NLP pipeline against AIDA[13] [12] on several English language documents. Comparison results are shown in Figure 6 on the log-log scale. In Figure 6a we can see that the text preprocessing stage of the Lithium pipeline is about 30,000-50,000 times faster compared to AIDA which is based on Stanford NLP NER [7]. The results for the disambiguation stage are shown in Figure 6b. The disambiguation stage for both the systems take a similar amount of time. However, AIDA fails to extract as many entities as evident in Figure 6c which shows that AIDA extracts 2.8 times fewer entities per $50kb$ of text. Finally, the disambiguation runtime per unique entity extracted of Lithium pipeline is about 3.5 times faster than AIDA as shown in Figure 6d. In conclusion, although AIDA entity disambiguation is fairly fast and robust, our system's runs significantly faster and is capable of extracting many more entities.

7.4 Comparison on demonstrative example

In order to explicitly demonstrate the benefits and expressiveness of our system, we also compare the results of our EDL system with Google Cloud NL API, OpenCalais and AIDA on the example that we discussed in Section 4.3. Figure 7 shows the disambiguation and linking results generated by our EDL system and the three other systems (Google NL

[12]We also analyzed AlchemyAPI (http://www.alchemyapi.com/resources) but it only processed a limited amount of text in a document and was not very stable on languages other than English.
[13]https://github.com/yago-naga/aida

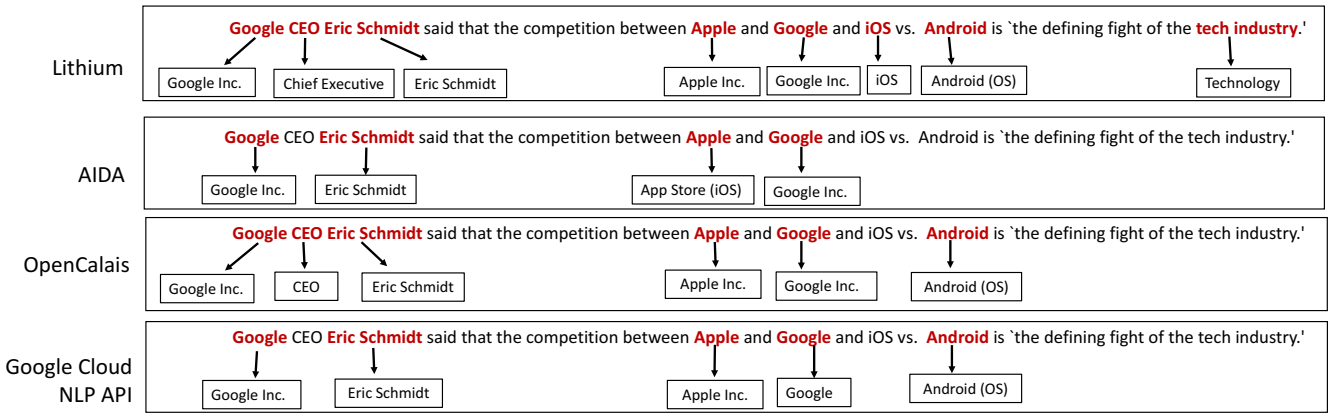


Figure 7: Comparison of the different systems on our demonstrative example

Cloud API, OpenCalais and AIDA) that we compare with. As evident, our EDL system disambiguates and links more entities correctly than the other 3 systems. All the other systems fail to disambiguate and link *iOS* and *tech industry*. In addition, AIDA incorrectly disambiguates *Apple*.

8 CONCLUSION AND FUTURE WORK

In this paper, we presented the Lithium EDL system that disambiguates and links entity mentions in text to their unique Freebase ids. Our EDL algorithm uses several context dependent and context independent features to disambiguate mentions to their respective entities. Moreover, it recognizes several types of entities in addition to named entities like people, places, organizations. In addition, our EDL system is language-agnostic and currently supports several languages including English, Arabic, Spanish, French, German, and Japanese. As a result, it is highly applicable to process real world text such as multi-lingual user generated content from social media in order to model user interests and expertise.

We compared our EDL system with several state-of-the-art systems and demonstrate that it has high throughput and is very lightweight. It can be run on an off-the-shelf commodity machine and scales easily to large datasets. Also, our experiments show that our EDL system extracts and correctly disambiguates about 75% more entities than existing state-of-the-art commercial systems such as Google NLP Cloud API and Open Calais and is significantly faster than some of them. In future, we plan to add support for several other languages to our EDL system once we have collected enough ground truth data for them. We also plan to migrate to Wikipedia as our KB. We will also compare our system's performance against several state-of-the-art systems on metrics such as precision, recall and f-score with respect to existing benchmarked datasets.

REFERENCES

[1] Prantik Bhattacharyya and Nemanja Spasojevic. 2017. Global Entity Ranking Across Multiple Languages. In *Proceedings of the 26th International Conference on World Wide Web*. to appear.

[2] Luka Bradesko, Janez Starc, and Stefano Pacifico. 2015. Isaac Bloomberg Meets Michael Bloomberg: Better EntityDisambiguation for the News. In *24th International Conference on World Wide Web*.

[3] Razvan C Bunescu and Marius Pasca. 2006. Using Encyclopedic Knowledge for Named entity Disambiguation.. In *EACL*. 9–16.

[4] Rudi L Cilibrasi and Paul MB Vitanyi. 2007. The google similarity distance. *IEEE Transactions on knowledge and data engineering* 19, 3 (2007), 370–383.

[5] Silviu Cucerzan. 2007. Large-Scale Named Entity Disambiguation Based on Wikipedia Data.. In *EMNLP-CoNLL*, Vol. 7. 708–716.

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

[7] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. In *43rd Annual Meeting on Association for Computational Linguistics*. 363–370.

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

[9] Tom Heath and Christian Bizer. 2011. Linked data: Evolving the web into a global data space. *Synthesis lectures on the semantic web: theory and technology* 1, 1 (2011), 1–136.

[10] Rada Mihalcea and Andras Csomai. 2007. Wikify!: Linking Documents to Encyclopedic Knowledge. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management (CIKM '07)*. 233–242.

[11] David Milne and Ian H. Witten. 2008. Learning to Link with Wikipedia. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management (CIKM '08)*. 509–518.

[12] Dat Ba Nguyen, Johannes Hoffart, Martin Theobald, and Gerhard Weikum. 2014. AIDA-light: High-Throughput Named-Entity Disambiguation.. In *LDOW*.

[13] Nemanja Spasojevic, Preeti Bhargava, and Guoning Hu. 2017. DAWT: Densely Annotated Wikipedia Texts across multiple languages. In *Proceedings of the 26th International Conference Companion on World Wide Web*. International World Wide Web Conferences Steering Committee, to appear.

[14] Nemanja Spasojevic, Prantik Bhattacharyya, and Adithya Rao. 2016. Mining half a billion topical experts across multiple social networks. *Social Network Analysis and Mining* 6, 1 (2016), 1–14.

[15] Nemanja Spasojevic, Jinyun Yan, Adithya Rao, and Prantik Bhattacharyya. 2014. LASTA: Large Scale Topic Assignment on Multiple Social Networks. In *Proc. of ACM Conference on Knowledge Discovery and Data Mining (KDD) (KDD '14)*.

[16] Mohamed Amir Yosef, Johannes Hoffart, Ilaria Bordino, Marc Spaniol, and Gerhard Weikum. 2011. Aida: An online tool for accurate disambiguation of named entities in text and tables. *Proceedings of the VLDB Endowment* 4, 12 (2011), 1450–1453.

[17] Mohamed Amir Yosef, Marc Spaniol, and Gerhard Weikum. AIDArabic: A named-entity disambiguation framework for Arabic text. In *The EMNLP 2014 Workshop on Arabic Natural Language Processing*. 187–195.