Named Entity Extraction and Knowledge Base Enhancement

Thesis
submitted in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Computer Science

Submitted by:
Priya Radhakrishnan
201050035
priya.r@research.iiit.ac.in

International Institute of Information Technology, Hyderabad
(Deemed to be University)
Hyderabad - 500 032, INDIA
April 2018
CERTIFICATE

It is certified that the work contained in this thesis, titled “Named Entity Extraction and Knowledge Base Enhancement” by Priya Radhakrishnan, has been carried out under my supervision and is not submitted elsewhere for a degree.

Date

Adviser: Prof. Vasudeva Varma

Date

Adviser: Dr. Manish Gupta
Dedicated to my father Late Dr. S. Radhakrishnan.
This image of Kim Clijsters with 2009 US Open trophy in one arm and her daughter Jada cradled in the other, went a long way in motivating me, then a new mother, exhausted and on the verge of giving up on technical aspirations. Thanks Kim!!

I thank my three teachers for helping me realize this thesis, Prof. Vasudeva Varma, my guide, Dr. Manish Gupta my co-guide, and Dr. Partha Pratim Talukdar my mentor at IISc, Bangalore. I thank VV Sir for showing belief in STEM abilities of girls, especially when the candidate in question is married and is a mother of two! I thank Manish for helping me reason out my thesis goal and work towards it in a disciplined manner. In any long distance race, the finishing lap is run with least energy but is the most watched lap of the race. Thanks Partha for helping me finish in style.

I thank two attitudes which made my research possible. First, the give-back-to-India attitude of this group of teachers who after earning PhDs from some of the finest academic institutions world-wide, decided to return to India and made IIIT, Hyderabad a reality. This attitude, epitomized by Dr. Anoop Namboodiri as he introduced me to research and IIIT, Hyderabad, has had a profound impact on me and the students of IIIT, Hyderabad. Thanks Anoop. Second, the attitude to do everything one can to help someone to learn and gain knowledge. This attitude, epitomized by my husband Hari and ably shared by my in-laws, sons and my mother, is the reason my research exists. Thanks Hari.

Finally, I would like to acknowledge my lab-mates at IRE lab in IIIT and MALL lab in IISc, hostel and malayalee friends and all teaching and non-teaching staff of IIIT, Hyderabad who helped me in my PhD journey.
Abstract

Past decade witnessed an explosive growth in the amount of unstructured data, especially in the public domain, mainly due to Web 2.0 and social media. This has created a need for applications that extract structured information from such noisy data. Automatic extraction of structured information from unstructured data is called information extraction. Structured information thus extracted is stored in a Knowledge Base. The knowledge base stores facts about entities like name, type and other attributes. On the one hand, the information extraction task utilizes the facts of the entity stored in knowledge base to refine the extraction process, while on the other, the facts extracted refines the knowledge base facts further. Thus knowledge base provides structure and guidance to the extraction task, and gets enhanced by the results of the extraction task. Here we see that the tasks of entity extraction and knowledge base enhancement are mutually dependent and mutually beneficial. Hence this thesis proposes methods to enhance both the tasks, in an effort to build a strong and sound named entity extraction system.

Documents typically talk about multiple named entities. All the named entities mentioned in the document are not equally important to the content of the document. Named entities that are important to the document are called salient named entities. In this thesis we propose a method to identify salient named entity of the document. Importance of a named entity can also be judged by understanding how the named entity is semantically related to other named entities mentioned in the document. We propose a method to identify the presence of such semantic relations within named entities. For example, semantic relation like attribute or category in a product title. Understanding the salience of a named entity and its semantic relations help the Named Entity Extraction task in extracting the important information from text, while filtering out unimportant information.

Performance of Named Entity Extraction methods depend on the size and structure of the context of the named entity mention in the text. While bigger size and better structure of context results in improved performance of the Named Entity Extraction, lower size and poor structure of context results in reduced performance. We propose three different Named Entity Extraction approaches tailored to the varying size and structure of the context in this thesis. The proposed methods perform on par with state-of-the-art methods with improved latency. Named Entity Extraction approaches that work on lesser context and poorer structure, increasingly depend on non-textual signal like global coherence of entities in the knowledge base.

Conventional EL performs well for popular entities but performs poorly for less popular (a.k.a tail) entities, because conventionally EL methods depend on richness of entity neighborhood in the Knowl-
edge Graph (KG). In this context ‘KB’ refers to Knowledge Base like Wikipedia and ‘KG’ refers to Knowledge Graph like Wikipedia Hyperlink Graph. Tail entities have sparse entity neighborhood and hence EL methods perform poorly on them. In this thesis we propose ELDEN, an EL system that overcomes the degree sparsity problem of tail entities. ELDEN enriches entity’s neighborhood in a KG by extracting high quality mentions of entity from a web corpus using Pointwise Mutual Information (PMI) measure. ELDEN outperforms state-of-the-art EL systems while achieving significant improvement in linking tail entities, achieving best results on CoNLL and TAC datasets. We follow up the discussion on ELDEN with a discussion on information retrieval task that is improved by use of KB entities. We propose a novel method for enhancing classification performance of research papers into ACM computer science categories using KB entities, both Wikipedia and Freebase entities.

All through this thesis we present five methods of improving Named Entity Extraction using Knowledgebases. We conclude the thesis by looking at how Knowledgebases can be improved with Named Entity Extraction. We review and analyze the main approaches of New Entity Identification (NEI) in Named Entity Extraction systems. We analyze the features and share insights from reproducing state-of-the-art results, suggesting future improvements.
## Contents

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction</td>
</tr>
<tr>
<td>1.1</td>
<td>Motivation</td>
</tr>
<tr>
<td>1.2</td>
<td>Problem Statement</td>
</tr>
<tr>
<td>1.3</td>
<td>Scope</td>
</tr>
<tr>
<td>1.4</td>
<td>Challenges</td>
</tr>
<tr>
<td>1.5</td>
<td>Contributions of the Thesis</td>
</tr>
<tr>
<td>1.6</td>
<td>Organization of the Thesis</td>
</tr>
<tr>
<td>2</td>
<td>Background and Related literature</td>
</tr>
<tr>
<td>2.1</td>
<td>Background</td>
</tr>
<tr>
<td>2.1.1</td>
<td>Entity Linking</td>
</tr>
<tr>
<td>2.1.2</td>
<td>Knowledge Base</td>
</tr>
<tr>
<td>2.1.3</td>
<td>Interdependence of Entity Linking and Knowledgebase Enhancement Tasks</td>
</tr>
<tr>
<td>2.2</td>
<td>Related Work</td>
</tr>
<tr>
<td>2.2.1</td>
<td>Named Entity Recognition</td>
</tr>
<tr>
<td>2.2.2</td>
<td>Named Entity Recognition in tweets</td>
</tr>
<tr>
<td>2.2.3</td>
<td>Salient Entities</td>
</tr>
<tr>
<td>2.2.4</td>
<td>Entity Attribute Recognition</td>
</tr>
<tr>
<td>2.2.5</td>
<td>Sequence prediction models used in NE Mention detection</td>
</tr>
<tr>
<td>2.2.6</td>
<td>Entity Linking</td>
</tr>
<tr>
<td>2.2.7</td>
<td>Entity Linking in Short Texts and Tweets</td>
</tr>
<tr>
<td>2.2.8</td>
<td>Knowledgebase Enhancement and Entity Linking</td>
</tr>
<tr>
<td>2.3</td>
<td>Conclusion</td>
</tr>
<tr>
<td>3</td>
<td>Overview of Approach</td>
</tr>
<tr>
<td>3.1</td>
<td>Identifying the Named Entity to link to Knowledgebase</td>
</tr>
<tr>
<td>3.2</td>
<td>Linking Named Entities to Knowledgebase</td>
</tr>
<tr>
<td>3.3</td>
<td>Knowledgebase Enhancement and Entity Linking</td>
</tr>
<tr>
<td>4</td>
<td>Identifying the Named Entities to Link</td>
</tr>
<tr>
<td>4.1</td>
<td>Salience of a Named Entity</td>
</tr>
<tr>
<td>4.2</td>
<td>Approach to identifying salience of Named Entities</td>
</tr>
<tr>
<td>4.3</td>
<td>Experiments on identifying salience of Named Entities</td>
</tr>
<tr>
<td>4.3.1</td>
<td>Dataset</td>
</tr>
<tr>
<td>4.3.2</td>
<td>Experiments</td>
</tr>
</tbody>
</table>
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1 Interdependence of EL and KB tasks.</td>
<td>3</td>
</tr>
<tr>
<td>1.2 Entity Disambiguation Difficulty</td>
<td>3</td>
</tr>
<tr>
<td>1.3 EL performance varies with varying context structure and length</td>
<td>7</td>
</tr>
<tr>
<td>1.4 Entity distribution by edge count in Wikipedia</td>
<td>8</td>
</tr>
<tr>
<td>2.1 NEE and KB enhancement are interdependent tasks</td>
<td>13</td>
</tr>
<tr>
<td>3.1 Two stages of NE attribute identification.</td>
<td>24</td>
</tr>
<tr>
<td>3.2 Tail Entity Linking</td>
<td>29</td>
</tr>
<tr>
<td>4.1 SNE of a tweet</td>
<td>33</td>
</tr>
<tr>
<td>4.2 Annotation Interface</td>
<td>35</td>
</tr>
<tr>
<td>4.3 NE_Type distribution in CWC15</td>
<td>37</td>
</tr>
<tr>
<td>4.4 SNEIT System Architecture</td>
<td>38</td>
</tr>
<tr>
<td>4.5 Performance in identifying B-SNE and I-SNE</td>
<td>41</td>
</tr>
<tr>
<td>4.6 Page rank vector creation</td>
<td>44</td>
</tr>
<tr>
<td>4.7 Execution time</td>
<td>46</td>
</tr>
<tr>
<td>4.8 &quot;Newer Model&quot; Feature on Amazon</td>
<td>50</td>
</tr>
<tr>
<td>5.1 Overview of EL system in documents</td>
<td>59</td>
</tr>
<tr>
<td>5.2 Overview of EL system in tweets</td>
<td>62</td>
</tr>
<tr>
<td>6.1 ELDEN approach</td>
<td>73</td>
</tr>
<tr>
<td>6.2 EL Performance on tail entities of CoNLL and TAC dataset</td>
<td>79</td>
</tr>
<tr>
<td>6.3 KG densification reduces false positives in EL of tail entities</td>
<td>82</td>
</tr>
<tr>
<td>6.4 Effect of entity’s web content on EL precision</td>
<td>83</td>
</tr>
<tr>
<td>6.5 Web content is good source of additional information for edge densification of entities</td>
<td>84</td>
</tr>
<tr>
<td>8.1 Representative of the Thresholding NEI approach</td>
<td>96</td>
</tr>
<tr>
<td>8.2 Representative of Supervised ML Method NEI Approach</td>
<td>97</td>
</tr>
<tr>
<td>8.3 NEI Accuracy</td>
<td>99</td>
</tr>
<tr>
<td>9.1 Interdependent tasks</td>
<td>102</td>
</tr>
<tr>
<td>9.2 Initial EL model</td>
<td>103</td>
</tr>
<tr>
<td>9.3 Present EL model</td>
<td>104</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

B.1 Edge count plot in Wikipedia KG showing tail effect. . . . . . . . . . . . . . . . . . . 108
# List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>4.1</strong> Inter annotator agreement scores</td>
<td>36</td>
</tr>
<tr>
<td><strong>4.2</strong> Features of labeler</td>
<td>39</td>
</tr>
<tr>
<td><strong>4.4</strong> NER Comparison</td>
<td>39</td>
</tr>
<tr>
<td><strong>4.5</strong> SNE Identification</td>
<td>39</td>
</tr>
<tr>
<td><strong>4.3</strong> CRF Features</td>
<td>40</td>
</tr>
<tr>
<td><strong>4.6</strong> Choice of KB</td>
<td>43</td>
</tr>
<tr>
<td><strong>4.7</strong> Choice of retrieval rank</td>
<td>43</td>
</tr>
<tr>
<td><strong>4.8</strong> Tweet Linker Performance with different classifiers</td>
<td>45</td>
</tr>
<tr>
<td><strong>4.9</strong> Evaluation with RepLab 2013 dataset</td>
<td>45</td>
</tr>
<tr>
<td><strong>4.10</strong> Comparison of SNEIT system performance with baseline</td>
<td>46</td>
</tr>
<tr>
<td><strong>4.11</strong> Semantic Relation of NE: Stage 1 results set 1</td>
<td>54</td>
</tr>
<tr>
<td><strong>4.12</strong> Semantic Relation of NE: Stage 1 results set 2</td>
<td>54</td>
</tr>
<tr>
<td><strong>4.13</strong> Semantic Relation of NE: Stage 2 results</td>
<td>55</td>
</tr>
<tr>
<td><strong>5.1</strong> TAC KB Entity Distribution</td>
<td>61</td>
</tr>
<tr>
<td><strong>5.2</strong> Results from EL on tweets</td>
<td>64</td>
</tr>
<tr>
<td><strong>6.1</strong> Notation used ELDEN approach</td>
<td>74</td>
</tr>
<tr>
<td><strong>6.2</strong> EL Features</td>
<td>75</td>
</tr>
<tr>
<td><strong>6.3</strong> Wikipedia KG processing details</td>
<td>77</td>
</tr>
<tr>
<td><strong>6.4</strong> ELDEN versus State-of-the-art EL systems</td>
<td>78</td>
</tr>
<tr>
<td><strong>6.5</strong> Evaluation dataset tail entity distribution</td>
<td>79</td>
</tr>
<tr>
<td><strong>6.6</strong> ELDEN ablation analysis</td>
<td>80</td>
</tr>
<tr>
<td><strong>6.7</strong> ELDEN Error Analysis</td>
<td>81</td>
</tr>
<tr>
<td><strong>7.1</strong> Performance Comparison with Baseline Methods</td>
<td>89</td>
</tr>
<tr>
<td><strong>7.2</strong> Performance Comparison with Different KBs</td>
<td>89</td>
</tr>
<tr>
<td><strong>8.1</strong> NEI Approaches grouping</td>
<td>95</td>
</tr>
<tr>
<td><strong>8.2</strong> NEI Performance comparison</td>
<td>99</td>
</tr>
<tr>
<td><strong>9.1</strong> Comparison of EL models presented in the thesis</td>
<td>104</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

While reading Natural language text, human readers often encounter unfamiliar terms. Readers typically refer to dictionaries and online encyclopedias like Wikipedia to obtain detailed information on the new or unfamiliar terms. When the unfamiliar term is a name like name of person or location or organization, Wikipedia is preferred over dictionary. Wikipedia, the free online encyclopedia, has opened the field of disambiguation of names. It contains articles on general entities like dog or science and specific entities like Donald Trump or London Bridge. Specific entities or entities with a specific name are referred to as Named Entities (NE).

The increasing use of Wikipedia for looking up NEs led to Wikification [?] which is resolving all words in a text to their corresponding Wikipedia articles. Here each article in Wikipedia is seen as one unit in a sense inventory which can be referred to by a program that resolves ambiguous names automatically. Sense inventory like Wikipedia is called Knowledge base (KB), as it provides the information about world’s entities. KB stores information on entities like entity name, attributes of the entity, entity’s semantic categories and relationships between entities. Wikification systems automatically identify concept appearing in a text document and links it to (or grounds it in) the corresponding article in Wikipedia. The task of linking concept text (also called textual entities) to entities in KB is called Entity Linking (EL).

Entity Linking system helps us look-up information about a entity when it is present in a KB. KB stores named entities and un-named entities. For example Wikipedia stores 4,413,000 entities out of which 1,575,966 are NEs [? ]. We see that sense inventories have more un-named entities and lesser number of named entities. In this thesis we look at ways to improve the number of NEs represented in the KB. Hence for the scope of this thesis, we restrict our attention to only NEs.

While KB stores the information about NEs, all the NEs of the world will not be present in the KB. In such cases where information about an NE is not present in the KB, the information is extracted from unstructured textual sources. The automatic extraction of structured information about NE such as entity name, type, relationships between entities, and attributes describing entities from unstructured sources is called Named Entity Extraction (NEE).

The field of study on automatic extraction of structured information such as entity names, entity type, relationships between entities, and attributes describing entities from unstructured sources is called Information Extraction. In this thesis we look at Information Extraction on Named Entities. Information extraction makes it possible to integrate the structured and unstructured data sources.

1.1 Motivation

Past decade has seen unprecedented growth in the amount of unstructured data. It has paved way to applications that can extract structured information from unstructured text. These new and emerging applications have lead to a frenzy of research and commercial activity on information extraction. As society becomes more data oriented with easy online access to both structured and unstructured data, new applications of structured information extraction come about.

Structured information extraction is useful in a diverse set of applications in enterprise, personal, scientific and Web-oriented areas [6, 8]. This includes news tracking, customer care, data cleaning and classified ads in enterprise. Personal information management (PIM) systems organize personal data like documents, emails, projects and people in a structured inter-linked format. The recent rise of the field of bio-informatics has broadened the scope of earlier extractions from named entities, to biological objects such as proteins and genes. Web oriented applications include citation databases, opinion databases, community websites, comparison shopping, Ad-placement on web pages and structured web searches. To address the needs of these diverse applications, the techniques of structured extraction are also evolving constantly.

1.2 Problem Statement

In this thesis, we present methods to extract structured information on named entities from unstructured texts and to enhance a KB. While the amount of data available in the structured format has increased in a steady rate, the amount of data in the unstructured format has seen unprecedented growth in the last two decades with the advent of Web 2.0. This has necessitated better ways to extract structured information from unstructured text.

Named Entity Extraction is generally referred to as Entity Linking in literature. It consists of two tasks, (i) detecting mention of the entity in the text and (ii) identifying the entity in the KB that represents the entity mentioned in text. These tasks as called Mention Detection and Disambiguation. Both mention detection and entity disambiguation depend on knowledge base for its successful completion. An enhanced KB helps achieve higher precision in EL. We depict these inter dependencies in Figure 9.1.

Mention Detection: While a document may talk about many entities, it does not make sense to extract information on all entities talked about. Our approach is to identify the NE that is central to the content of the document. The NE that is central to the content of the document is called the salient NE of the document. So if we extract the information on the salient NEs, it would represent the information
of the document. We further filter the NEs mentioned in a document based on how a NE is semantically related to other NEs mentioned in the document. Thus we refine the task of mention detection using information on whether the NE is the salient entity of the document and if the NE is semantically related to other NEs mentioned in the document.

*Disambiguation*: Entity disambiguation is a difficult and non-trivial task. Consider the following example sentence in Figure 1.2. There are three NEs mentioned in this sentence namely *India*, *MOM* and *mars*.

The first mention *India* refers to the country *India*. However this could also be referred as Bharat or Hindustan. This phenomenon of the same entity (*India*) being referred by different names is called *synonymy*. The second mention *MOM* could refer to the entities *mother*, *MOM* (the popular sitcom) or *Mars Orbiter Mission*. This phenomenon of the same mention referring to many entities is called *polysemy*. Both synonymy and polysemy are prominent issues in Entity Disambiguation. The third mention *mars* refers to the planet *Mars*. The fact that first mention refers to India the country

---

2We use *italics* to denote textual mentions and *typewriter* to indicate an entity in KB.
and third mention refers to Mars the planet helps disambiguate MOM as Mars Orbiter Mission a space probe orbiting Mars launched by the Indian Space Research Organization. Thus we see that using the contextual evidences (context refers to adjacent words i.e. k words to the left and right of the word to be disambiguated) and global evidences (KB relations between referent entities of the mentions) we disambiguate the NE. Thus we can formulate NE disambiguation problem as a combination of function of contextual evidences and function of global evidences.

Given a KB, usually in the form of a graph, and a document $D$ with mentions marked up, (usually through a named entity recognition process), entity linking aims at assigning unique entities from the KB to those mentions, whenever appropriate. More precisely:

**Definition 1** Entity Linking (EL). Given a set of mentions $M = \{m_1, \ldots, m_n\}$ in a document $D$, and a knowledge base (KB) whose entity set is $E$, the problem of entity linking is to find an assignment $\Lambda : M \rightarrow E$.

In this definition, finding the set of mentions $M$ is the mention detection task and finding the assignment $\Lambda$ is the entity disambiguation task. Goal of disambiguation is to find the best assignment $\Lambda$ that is contextually compatible with mentions in $M$, and has maximum coherence. Formally, The solution is an assignment $\Lambda^*$ maximizing the following objective function:

$$\Lambda^* = \arg \max_{\Lambda} \left( \sum_{i=1}^{n} \phi(m_i, e_i) + \Psi(\Lambda) \right)$$ (1.1)

in which $\phi(m_i, e_i)$ measures the contextual compatibility of mention $m_i$ and entity $e_i$, and $\Psi(\Lambda)$ measures the global coherence of assignment $\Lambda$. Contextual compatibility measure $\phi(m_i, e_i)$ is typically obtained by combining the prior probability and context similarity. Global coherence $\Psi(\Lambda)$ measures the coherence among entities in an assignment $\Lambda$ with $\psi(e, D)$, the semantic relatedness measure. For mention $m$, if $C$ are the candidate entities for $m$, the assignment is

$$\Lambda = \arg \max_{e_i \in C} \left( \phi(m_i, e_i) + \psi(e_i, D) \right)$$ (1.2)

Semantic relatedness of an entity $e_i$ to document $D$, $\psi(e_i, D)$ can be computed given the set of entities $E_D$ representing $D$, from semantic relatedness between entities, $SR(e_i, e_j)$. This is calculated in many ways including the popular TAGME method (Eq. 1.3) and 2-step assignment method as in Equation 1.4.

$$\psi(e_i, D) = \sum_{\text{mention}} \sum_{e_j \in C} SR(e_i, e_j)$$ (1.3)

In the TAGME method [?], after all the mentions and all the candidates of each mention are identified, $SR$ of a candidate with every other candidate for all the mentions identified is calculated and summarized.

$$\psi(e_i, D) = \sum_{e_j \in k} SR(e_i, e_j)$$ (1.4)
In 2-step assignment method [? ? ]. Unambiguous mentions are assigned in step-1 and ambiguous mentions are disambiguated in step-2. A mention is said to be unambiguous if there is only one entity in the KB associated with it. We initialize the set $E_D$ by assigning the referent entities of all unambiguous mentions in $M$. Let this initial assignment be $A$. $SR$ of each candidate entity with assigned entity is summarized to find $\psi(e_i, D)$ as in Equation 1.4. In this thesis we explore various implementations of $SR$ and see how improvements to them results in improved EL performance.

EL process is a well refined list look-up task. Hence it’s obvious limitation is the completeness of the list. While creating a list of all possible NEs that humans might be interested in, is possible, it is impossible to have it human edited. The nearest possible list of all possible NEs that is human edited, is the list of NEs in Wikipedia. We use the Wikipedia NE list for distant supervision, in learning our wish list of all possible NEs. We learn embeddings of all entities and evaluate them using embeddings of Wikipedia entities.

1.3 Scope

Research in information extraction involves techniques from four fields namely machine learning, databases, information retrieval and computational linguistics. In this thesis the work involved is mainly concerned with information retrieval and machine learning, and to a lesser extent with databases and computational linguistics.

The word or group of words that uniquely identify an entity is called the name of the entity. A KB stores information on entities with unique name such as *Donald_Trump* and *London_Bridge* and entities without a unique name indicating a concept or class, such as *Dog* and *Archaeology*. We refer to entity with a unique name as Named Entity (NE). This thesis is about Named Entities alone. We have used the terms ‘entity’ and ‘named entity’ interchangeably in the rest of the thesis. Reader is advised to interpret both as named entity.

Identifying NEs in text is a well established research area. Systems that identify NEs in text are called Named Entity Recognizers (NER). NERs have achieved near human perfection (93.96% on MUC7 dataset) in English on structured documents. However in semi-structured and unstructured documents the accuracy is low. Further when used in application like extracting name and type of NE for populating / enhancing a KB, the performance deteriorates. The scope of the thesis is restricted to enhancing Named Entity Extraction using NER output. Enhancing NER output or tagging of sentence, is beyond the scope of this work. Also we restrict the scope of this thesis to English language text alone.

Information extraction research is typically described in terms of extraction task, techniques used for extraction, variety of input sources exploited and the type of output produced. The structures extracted by our work are named entities and attributes. This does not cover relationship between entities, list, tables etc. The unstructured text sources explored by our study spans all the three variations of document structure namely structured as in documents (webpages, e-mails, weblogs, discussion forums) semi-structured (product description, web-postings, tweets) and unstructured (search queries).
Input resources: The input resources include structured knowledge bases (TAC KB, Wikipedia, DBpedia and Freebase), labeled unstructured data (Google Concept Dataset) and linguistic tags (tags from Stanford NER parser, Ritter NLP toolkit etc). The methods of extraction employed in this study include heuristic (rule based), statistical unsupervised (including classifiers and Deep Neural Networks) and statistical with supervision as in manually trained from examples (CRF taggers trained on labeled dataset). Finally the output from our study is unstructured text annotated with Named Entities in a knowledge base, which is ready for consumption of down the line information extraction tasks like sentiment analysis, question answering systems and so on.

Textual scope : Named Entity Extraction presented in this thesis is based on how the entity is presented in the document under study. Here the premise or underlying assumption is that the NE is explicitly explained in the document and its information can be efficiently extracted from the textual content.

1.4 Challenges

Named Entity Extraction comes with its own share of issues and challenges. While the problems like acronym (U.S and United States of America), synonym (Briton and United Kingdom) are established in the field of Information Retrieval (IR), problems like identifying the salient of the NE mentions of a document is specific to Named Entity Extraction.

C1 Salience of Entities: While the concept of ‘relevance’ is well known in the IR world, the concept of ‘salience’ is relatively less known. Salience is a function of the structure of text, and indirectly a function of the intention of the author [?]. Another definition of salience is “Salience is a measure of the relative prominence of objects in discourse: objects with high salience are the focus of attention; those with low salience are at the periphery”. Salience is not same as relevance. While relevance of a document is a function of information need (of a seeker), salience is a function of document content. Thus salience does not change with information need or seeker. Hence salient NE is a better candidate for information extraction than relevant NE. Further information extraction from sources like tweets mostly involve tapping public opinion or sentiment. Here extracting information about the document’s salient entity is required rather than knowing the analyzer’s information requirement.

C2 Context as aid: In most named entity extraction applications, the context of the NE mention is biggest aid in disambiguation of NE. However with document type moving from web-pages and email to tweet and queries, the context is reducing. Figure 1.3 shows increasing context size and structure of the document on the x-axis and difficulty of EL on that document on the y-axis. Here we see that when the document is well structured and has abundant context, EL performance is good. When the document has poor structure and context, EL performance is poor. Short text like tweets occupy the median bar. Extracting information from tweets is challenging in many ways. Tweets are short in length (maximum 140 characters), resulting in ambiguity and reduced context. They are also dynamic, context-dependent and less grammatical than longer posts. Especially in extracting Named Entities
(NE) from tweets, the use of unorthodox capitalization leads to significant drop of recall compared to conventional text [? ]. As tweet content is not curated (as done by editors in newswire), tweets are largely superfluous impacting information extraction performance. Thus analyzing tweet content is an increasingly interesting research problem [? ].

C3 Limited ground truth for training and evaluation: Machine learning models for NER and EL for tweets suffer from the lack of annotated training data. In a survey of NERs for tweets, Derczynski et al. [? ] observe that creating more human-annotated training corpora of microblog content will allow better algorithm adaptation and parameter tuning to the specifics of this genre. For example, there are currently fewer than 10,000 tweets annotated with named entity types, which is far from sufficient.

Existing studies on salience estimation use search log and proprietary (New York Times) corpus to estimate salience of entities. We created dataset capturing the salience of NEs in tweets containing 10,938 tweets and their Salient NEs. As this is a human annotated dataset, scaling it to the tune of millions is not possible.

In evaluating NE attributes in our experiments, we have crawled websites of e-commerce portals. This information scraping is also limited by the number of queries allowed by the e-commerce websites.

C4 Poor performance of Entity Linking on tail entities. EL performs well for popular entities but performs poorly for less popular (a.k.a tail) entities, because conventional EL methods depend on edges of an entity to other entities in knowledge graph, for linking an entity. Tail entities have lesser edges and hence EL methods perform poorly on them. Figure B.1 shows degree (edge count) vs. number of entities having that degree in the KG, in log-log scale. We see that few entities have many edges and many entities have few edges in Wikipedia. For example, new articles for named entities in Wikipedia

---

3In this graph, some points are appearing piled up, due to rounding. Datapoints like (59,1044), (58,1020), (57,1129), (56,1118), (55,1204) and (54,1228) map to 3.0 on x-axis.
Figure 1.4: Plot of edge count versus number of entities in Wikipedia with that edge count in log-log scale. As edge count increases, number of entities decreases, leading to few entities with many edges (head entities) and many entities with few edges (tail entities), showing the long-tail effect.

are first created as stub pages. These pages lack the desired encyclopedic coverage and connections to related entities. Thus articles start as tail entities in the KB. As their prominence (or popularity) grows in the public domain, more content gets added to their Wikipedia page and the hyperlinks in their article also increases, leading to more edges to the entity in the KG. Entities that have many edges are called head entities and entities that have less number of edges are called tail entities. Head entities are well connected to other entities in KB. Tail entities are less popular and are not as well connected to other entities in KB as head (or popular) entities. As the entity’s prominence increases in public domain, the article also gets richer in content and connections.

The number of tail entities in a KG is many folds larger than the number of head entities. In Figure B.1 we find that entities with 500 edges or less make up more than 90% of Wikipedia. Also state that the main difficulty in EL is in linking tail-entities. Thus linking tail entities accurately is critical to EL performance.

1.5 Contributions of the Thesis

In this thesis we address the task of enhancing named entity extraction and enriching the structured knowledge source enabling it. Starting from text we build a semantically enriched text by automatically detecting and disambiguating entity mentions to external knowledge resources. We leverage the context and structured attributes from disambiguated entities to improve named entity extraction. We now detail the main contributions of this dissertation:

1. **Salience of named entities** We introduce salience modelling and using salience of named entities in a document in choosing the mentions for named entity extraction. While the conventional EL
methods linked all the entity mentions, we propose a salience based filtering of the mentions. We prove that EL done on salient named entities gives higher semantic enrichment and relevance.

We model the task of salient named entity identification as a supervised machine learning task, and achieve an F-measure of 0.63. We evaluate our system with a standard tweet filtering task dataset as well as with other salient entity detection systems. Our method performs better than median accuracy of the tweet filtering task submissions, and outperforms two of the three salient entity detection methods.

2. Semantic attribute or category of a named entity We propose the novel problem of predicting the predecessor version of a given query product version. In the approach we see how one NE is predicted as the semantic attribute or category of another when both the NEs occur in a product title. We refer to the extracted attribute as ‘Semantic Attribute of NE’ in thesis as in terms of Knowledge Graph, each NE is a node and each edge is a semantic attribute (i.e relation) of the node (or NE). We model the prediction as a sequence labeling problem which uses the context and structural attributes of the NE.

We present a two-stage approach to solve the problem: (1) parsing the product title, and (2) predicting the predecessor version given all candidates. Experiments on a dataset crawled from Amazon show that our methods achieve a precision of 88% for the first stage, and a precision of 53% for the second stage.

3. Entity Linking solution tailored to document structure and context length Figure 1.3 shows how difficulty to link entities varies with context size and structure of the document. Standard EL methods report good performance when document is well structured and has abundant context. For document with poor structure and context, EL performance reported is poor. So we suggest three different solutions catered to the structure and context length of the document. When context length is high, our method uses the contextual evidence more i.e $\phi(m_{i,e_i})$ of objective function 6.1 and when context length is low, we use global coherence more i.e $\Psi(\Lambda)$ of objective function 6.1.

Our EL solution for documents was tested at the Text Analysis Conference (TAC 2014). It models the problem as a supervised classification problem and proposes a four stage solution. On development dataset, we achieved Precision (P) of 0.788 and Recall (R) of 0.19, while that of best participant was P of 0.717 and R of 0.642, by the strong typed link match metric. Our EL solution for tweets was tested on the manually annotated dataset provided by Named Entity Extraction and Linking (NEEL) Challenge 2014, and the obtained results are on par with the state-of-the-art methods with F1 of 0.52. Our EL solution for linking search queries was tested at the Entity Recognition and Disambiguation Challenge at SIGIR 2014. It performed with an F1 of 0.53 and was judged as system with lowest latency.

4. Entity Linking of tail entities The number of tail entities in a KB is many folds larger than the number of head entities. Hence linking tail entities accurately is critical to EL performance. In
this thesis we address the edge sparsity problem of tail entities by KB densification. Our solution (ELDEN) increases the number of edges to tail entities using evidences of entity from a web corpus.

ELDEN performs on par with state-of-the-art EL methods while it is most effective in case of tail entities. It presents the first ever comparison of performance of state-of-the-art EL systems in linking tail entities. ELDEN uses mentions of entities from a web corpus refined with PMI measure, to improve EL performance. As it uses a corpus based measure (PMI), it is general purpose and may be applied to improve EL involving any KB.

5. **Identifying new entities to add to Knowledgebase** Though identifying missing entities of a KB or identifying potential entities to be added to a KB is a critical step in the automatic construction and maintenance of KB, very little formal research has been conducted in this direction. We survey the literature on this problem, review and analyze the main approaches of New Entity Identification (NEI) in EL systems and the features used thereof. We re-implement representative algorithms from both NEI approach types to get insights from reproducing state-of-the-art results and identifying improvements.

### 1.6 Organization of the Thesis

- **Chapter 2** provides a brief background of the component tasks of NEE in the first part, where we acquaint the reader with terminology used in rest of the thesis. The second part details the related work in the areas of Named Entity Recognition, Entity Linking, Entity Attribute Extraction, Entity Salience Identification, Tail Entity Linking and Knowledgebase Enhancement. For each of the category, we present a the top few milestone works in that area, in a way that reader can appreciate the emergence of the field and appreciate the contribution of this thesis.

- **Chapter 3** presents a brief overview of the thesis approach so that it can be compared with other state-of-the-art approaches detailed in related work chapter.

The following chapters describe the thesis approach in detail. In particular, we describe various details of the approach bringing out how our method compares with state-of-the-art methods in approach and results.

- **Chapter 4** proposes solutions to the problem of identifying the salient NE of a document (Contribution 1). It presents experiments on extracting semantic category of NEs specifically from product titles (Contribution 2).

- **Chapter 5** presents enrichment applied to the task of entity linking. It details the various approaches to linking entities across various levels of text structure, from structured documents to less structured tweets and search queries (Contribution 3).
• Chapter 6 explores inter-dependence of method of knowledgebase enhancement and EL. It explains how KB enhancement improves EL performance, by means of experiments on tail entities (Contribution 4).

• Chapter 7 looks at application of linked entity in improving an information retrieval application namely Document categorization. It explains how classification performance of research papers into ACM computer science categories can be enhanced using KB entities.

• Chapter 8 In this chapter we see how EL can lead to KB Enhancement (Contribution 5). We analyze many ways of adding new entities to KB i.e. enhancing entities in KB using EL. We draw conclusions of best features and methods for NEI.

• Chapter 9 summarizes the contributions made in this thesis and discusses possible future research directions in the area.
Chapter 2

Background and Related literature

In this chapter we first present a brief introduction to named entity extraction and knowledge base enhancement tasks, as background knowledge. We then review related literature on the research problems discussed and improved in this thesis. Named Entity Extraction (NEE) is generally referred as Entity Linking (EL) in literature. We use the terms interchangeably in this thesis.

2.1 Background

2.1.1 Entity Linking

The word linking means ‘making, forming or suggesting a connection between two things’. Entity Linking is linking a named entity appearing in text to its entry in the Knowledge Base (KB). This includes two steps (i) identifying the named entity and (ii) identifying the knowledge base entry.

Identifying the named entity: Named Entity (NE) identification is determining the link-able phrases in a text. Though Wikification proposed by Mihalcea et al. [7] linked every term in text to pages in Wikipedia, Bunescu and Pasca [8] suggested linking only NEs to Wikipedia. Further studies like [9] have found that linking NEs is more effective or helpful to the reader compared to linking all entities. In this thesis we consider linking only NEs. Thus NE detection is identifying the phrases representing NE in the text, that can be linked to entities in the KB.

Identifying the knowledge base entry or Entity Disambiguation: Disambiguation is identifying the candidate entities from a knowledge base and selecting the entity to link the NE. This selection is based on local or contextual factors and global factors (Please refer Section 1.2 Equation 6.1). The availability of contextual factors help in disambiguation, while the lack of it also hinders the disambiguation performance as seen in Figure 1.3. Global factor is primarily the prior probability or importance of a KB entity. These factors however can favor frequently occurring or popular entities. In order to prevent prior probability biasing against rarely occurring entities, we use factors like coherence among NEs in the context, captured by semantic relatedness between entities $SR(e_i, e_j)$. 


2.1.2 Knowledge Base

Knowledge Base (KB) is a fundamental component of Entity Linking (EL) task. Wikipedia is used as the KB in majority of academic research in EL. This is because Wikipedia is large and freely available online encyclopedia. Other KBs used in EL studies include YAGO, DBpedia, Freebase, ReadTheWeb, KnowItAll and Probase. Many of them are built with Wikipedia along with other sources of entities. An entity in a KB has a name (a canonicalized name) a semantic category (type information), attributes and relationships to other the entities in KB. Hence creating or enhancing the KB involves gathering these information about the entities.

2.1.3 Interdependence of Entity Linking and Knowledgebase Enhancement Tasks

Canonicalization of entity names is mapping of various surface forms of an entity name to a single name (a.k.a canonical name). KB stores canonical name of the entity. Canonicalization is done in NE mention detection to tackle synonymy. Thus NE mention detection and therefore EL is critical task for KB creation and enhancement, as the output of EL is the input of KB. In order to disambiguate and link a NE, the EL task checks the representation of NE in the KB. Thus output of KB is input to EL too. Thus these tasks are interdependent like yin-yang. Semantic attribute of NE like Category or type information of an entity is determined by the entity’s attributes and grouping with similar entities. Category information helps place the entity KB tree which helps infer more information about the entity. Category information helps disambiguation of entity.
2.2 Related Work

Named Entity Extraction and Knowledge base enhancement consists of two tasks, (i) Identifying the NE to be extracted in the text and (ii) Linking or grounding it in a entry in the knowledge base. Standard named entity recognizer identifies words and phrases in the given text that are candidate to be NEs. From the candidate NEs, the NE to extract is chosen based on salience of the NE and probability of NE to be semantically related to other NEs in the given text. KB stores aspects of NE like canonicalized name, type and relationship with other entities. These are established by linking the chosen NE to the KB entity. In this section we look at the related literature starting from state-of-the-art methods for NERs, selecting salient NE and linking NEs to KB. We also look at literature of semantic category identification and KB enhancement that we discuss in this thesis.

2.2.1 Named Entity Recognition

A Named Entity (NE) is a word or phrase that clearly identifies one item from a set of items that have similar attributes [?]. In the expression named entity, the word named restricts the scope of entities that have one or many rigid designators that stand for a referent. Named Entity Recognition (NER) is the process of locating the word or phrase that references a particular entity within the text.

Research in NER has a twenty year old history, starting in Message Understanding [?]. The initial definition included recognizing entity names (people and organizations), place names, temporal expressions and numerical expressions. More fine grained classification is person, organization, location, product, art, event, building etc. proposed in 2006 [?].

The methods used for NER include supervised, semi-supervised and unsupervised methods. The state-of-the-art in NER is Stanford NER [?] module. Stanford NER is also known as CRFClassifier. The software provides a general implementation of (arbitrary order) linear chain Conditional Random Field (CRF) sequence models.

As we discussed in Section 1.3 we restrict the the scope of the thesis to NER output. We have considered NER as black box, and enhancing NER output or tagging of sentence, is beyond the scope of this work.

2.2.2 Named Entity Recognition in tweets

In a survey of NERs for tweets, Derczynski et al. [?] observe that Twitter-specific NER is difficult due to lack of sufficient context and good human-annotated corpus coverage of distinct named entity types. They also report that the highest achieved F1 score on NER in tweets is only 0.40[14].

The kinds of entities encountered in microblog corpora are somewhat different from those in typical text. For people entities, text like news corpora talk about politicians, business leaders and journalists, tweets talk about sportsmen, actors, TV characters and names of personal friends. Microblog corpora

[1] NER performance over the gold part of the UMBC dataset
shares only celebrities as common type with text in people entities. For location entities, news text mentions countries, rivers, cities and places with administrative function (parliaments, embassies). Tweets on the other hand discuss restaurants, bars, local landmarks, and sometimes cities. The fact that entities occurring in tweets are different from those in newswire makes it hard for the systems to tag them correctly. Further tweets have reduced context (maximum 140 characters). They are dynamic in nature and are less grammatical than longer posts. Further, in extracting NEs from tweets, the use of unorthodox capitalization leads to significant drop of recall compared to conventional text.

The first system that is open source and available for tweet NER is ANNI from GATE version 8, which uses gazetteer-based look-ups and finite state machines to identify and type NEs in newswire text. This was followed by the Stanford NER system enhanced with tweet training. This was done by training on news and tweet corpus. In 2011, Ritter et al. gave the thus far best performing system for NER on tweets. This is an open source system, which take a pipeline approach performing first tokenisation and POS tagging before using topic models to find NEs, reaching 83.6% F1 measure on the Ritter dataset.

Ritter et al. in their work, model the tweet NER problem as problem of segmenting tweets and classifying the segments into entity types. This work proposes a distantly supervised approach based on LabeledLDA and shows that it significantly outperforms Stanford NER. Developed in parallel to this work, Gimpel et al. built a POS tagger for tweets using 20 coarse-grained tags. In a survey of NERs for tweets, Derczynski et al. observe that Twitter-specific NER is difficult due to lack of sufficient context and good human-annotated corpus coverage of distinct named entity types. They also report that the highest achieved F1 score on NER in tweets is only 0.40 (NER performance over the gold part of the UMBC dataset).

In our experiments we overcome the low recall on tweet NERs by combining the outputs from multiple tweet NERs as discussed in Section 5.4.

2.2.3 Salient Entities

In their work on identifying salient entities in web pages, Gamon et al. assign salience scores to entities based on their centrality to the page content. Gillick and Dunietz extract salience specific features from the abstract of a document, on the assumption that salient entities are mentioned in it. Both the works focus on web documents, for which high-quality NLP tools are available. Our approach (SNEIT) focuses on tweets. Salience as a concept has received little attention in information retrieval and knowledge discovery. Most of the earlier works consider linguistic salience whereas we concentrate on entity salience. The linguistic salience of a text segment characterizes its attractiveness from a linguistic viewpoint. Its determinants can be phonetic, morphological, lexical, syntactic, semantic, rhetoric or pragmatic. In this thesis our emphasis is on entity salience, where we look at salience of an entity as the entity’s capacity to attract or catch an observers attention.

Meij et al. are the first to propose ‘whole tweet’ entity linking, where they identify concepts in the tweet and link them to Wikipedia entry. Their dataset is a concept disambiguated tweet
dataset, as it contains both tweet and the corresponding Wikipedia article, which are the concepts of these tweets. Their approach involves ranking followed by supervised classification to find the tweet’s concepts. SNEIT approach uses a similar supervised approach. The difference between the two systems is that Meij et al. try to estimate salience by ranking, whereas the proposed approach uses a sequence labeler for the task. Guo et al. look at NE detection and disambiguation as a single end-to-end task. However, it is not about linking the tweet to most prominent (and mostly single) entity, which is the case in whole tweet linking. In their work, Guo et al. utilize multiple tweets instead of a single tweet.

Labeling entity salience Gamon et al. use behavioral signals from web users as a proxy for salience annotation. Mining a web search log and click log from a commercial search engine, they assess the relevance and salience of an entity with respect to a URL. Gillick and Dunietz automatically generate salience labels for an existing corpus of document, abstract pairs using the assumption that the salient entities will be mentioned in the abstract. So they identify and align the entities in each text. Our method is similar to that of Gillick and Dunietz, in identifying and aligning SNE across an image and tweet. However annotation is done manually in our case whereas it is done automatically in Gillick and Dunietz method. Manual annotation was done by Deschacht et al. in the study of salience of entities to predict the probability of salient entity appearing in the image accompanying the text. To test their system, they annotated 900 image-text pairs of the YahooNews dataset. For every text-image pair one human annotator has selected the entities that appear both in the text and in the image and sorted these based on their perceived importance in the image. Annotation of CWC15 dataset for SNEIT was also done these lines.

2.2.4 Entity Attribute Recognition

Entity Attribute Recognition or Extraction is the problem of identifying the values for one or more attributes of an entity. Extracting attribute values and relationships from web tables is a well researched problem. Yakout et al. explain how web tables can be augmented to gather entity attribute information. There are similar experiments on product data by Mauge et al., Ghani et al. and Qui et al. Putthividhya presented a NER system for extracting product attributes and values from product title listings.

In this thesis we explore entity attribute extraction to understand how identifying an NE as an attribute or semantic category of another NE, helps in NE extraction. These experiments are related to research on (1) extracting attribute values for product entities, and (2) identifying related entities.

Mauge et al. structure items into descriptive properties using a two-step method (unsupervised property discovery and extraction, and supervised property synonym discovery). They mine this data from product descriptions. Raju et al. perform the task of automatically discovering attributes of products from text descriptions using an unsupervised approach. Ghani et al. present a semi-supervised co-EM algorithm for attribute-value entity extraction from product descriptions. Unlike these approaches, we aim at extracting information from product titles only. Putthividhya et al. present
a named entity recognition (NER) system for extracting product attributes and values from listing titles. However, they focus on clothing and shoe categories only and design methods specific to those domains. None of these works focus on extracting the version information from the product listing titles, which is the focus of the first stage of the proposed work.

Relationship extraction between entities has been studied recently \[?\]. However, all of these works \[?\] focus on semantic relation between entities, while the second stage of the proposed work deals with identifying temporally related product entities. To understand the relationship between statistically related entities, Fang et al. \[?\] propose the problem of entity relationship explanation. The temporal relationship discovered by the proposed work is self-explanatory.

2.2.5 Sequence prediction models used in NE Mention detection

Models used for predicting NEs in text can be broadly classified into supervised, semi-supervised and unsupervised models \[?\]. We present a brief overview of the models and present top few milestone works in it, in a way that reader can appreciate the emergence of the field and appreciate the contribution of this thesis.

Supervised Learning Algoritms include Hidden Markov Model (HMM), Maximum Entropy Models, Support Vector Machines and Conditional Random Field (CRF).

Hidden Markov Models (HMM): HMM models the NE prediction task as finding the most likely sequence of name-classes (NC) given a sequence of words (W):

\[ \max Pr(\text{NC}|W) \quad (2.1) \]

HMM is a generative model, i.e. it tries to generate the data, sequences of words W, and labels NC from distribution parameters.

\[ Pr(\text{NC}|W) = \frac{Pr(W, \text{NC})}{Pr(W)} \quad (2.2) \]

\( Pr(W, \text{NC}) \) is maximized using Viterbi algorithm over all possible name-class assignments. Zhou and Su \[?\] modified this model by using mutual information. They reported an accuracy of 96.6% on MUC-6 data and 94.1% on MUC-7 data.

Maximum Entropy based Model (MEM): Unlike HMM, MEM is a discriminative model. Given a set of features and training data, the model directly learns the weight for discriminative features for classification. In Maximum entropy models, objective is to maximize the entropy of the data, so as to generalize as much as possible for the training data. Maximizing the entropy ensures that for every feature \( g_i(\text{NC, W}) \), the expected value of \( g_i \), according to M.E. model will be equal to empirical expectation of \( g_i \) in the training corpus. This was used by Borthwick \[?\], Curran and Clark \[?\] who used the Viterbi and softmax approach to formulate the probability \( Pr(\text{NC}/W) \).

SVM Based Models: Support Vector Machine was first introduced by Cortes and Vapnik \[?\]. McNamara and Mayfield \[?\] use SVM to model NER as binary decision problem. They use 8 binary classifiers to decide if the word belongs to one of the 8 classes, i.e. B- Beginning, I- Inside tag for
Conditional Random Fields (CRF) Conditional random field were introduced by Lafferty et al. \cite{8}. CRFs define the conditional probability of a state sequence given an input sequence to be

\[
P(s|o) = \frac{1}{Z} \exp \left(\sum_{t=1}^{T} \lambda_k f_k(s_{t-1}, s_t, o, t)\right)
\]

where Z is the normalization factor obtained by marginalizing over all state sequences, \( f_k(s_{t-1}, s_t, o, t) \) is an arbitrary feature function and \( \lambda_k \) is the learned weight for each feature function. McCallum and Li \cite{9} proposed a feature induction method for CRF in NE. They achieved an accuracy of 84.04% on experiments performed on CoNLL 2003 shared task data.

Semisupervised Learning Algorithms Semi-supervised learning algorithms usually starts with small amount of annotated corpus as seed dataset. They create the initial hypothesis or classifiers with large amount of unannotated corpus. With each iteration, more annotations are generated and stored until a certain threshold occurs to stop the iterations.

Carreras et al. \cite{10} have modeled the Named entity identification task as sequence labeling problem through BIO labeling scheme and three binary classifiers. The binary AdaBoost is used with confidence rated predictions as learning algorithm for the classifier.

Unsupervised Learning Algorithms As supervised learning requires a robust set of features and large annotated corpus, researchers explored unsupervised methods for NER. Etzioni et al. \cite{11} proposed KNOWITALL which uses 8 domain independent extraction patterns to generate candidate facts. The plausibility of the candidate facts it extracts are tested using pointwise mutual information (PMI) computed using large web text as corpus. Latest trend in these algorithms is where the features and their weights are learned automatically. These algorithms constitute deep learning. The co-training algorithm (DL-CoTrain) proposed by Collins and Singer \cite{12} to learn a decision list (DL). The DL-CoTrain algorithm induces decision lists for multiple classes starting from user-provided content seed features for each class.

Deep learning is set of machine learning algorithms that attempt to learn layered model of inputs, commonly know as neural nets. Artificial neural networks consists of neurons. Each neuron is a perceptron with weighted inputs. The network of neurons follows a layered architecture. There are connections only among consecutive layers.

Primary computation node in neural network is a perceptron. Perceptron does a supervised classification of an input into one of the several possible non-binary outputs. It is a linear classifier that makes prediction using weighted sum of input feature vector. Mathematically, perceptron can be written as

\[
y = f(z)
\]

where

\[
z = \sum w_i x_i = W^TX
\]
If \( f \) is unity function then a perceptron is equivalent to a linear regression model. If \( f \) is sigmoid function then the perceptron acts as logistic regression. Back propagation algorithm \([?\ ]\) is used to train a multi-layer feed forward network. Basic idea of back propagation algorithm is to minimize the error with respect to input by propagating error adjustments on the weight of the network. Back propagation algorithm uses gradient decent method that calculates the squared error function with respect to the weights of the network. The squared error function is

\[
E = \frac{1}{2} \sum_{n \in \text{training}} (t^n - y^n)^2
\]  

(2.6)

where \( t = \) target output and \( y = \) actual output of the output node. Now differentiating the above error function gives the gradient of error that is propagated back.

\[
\frac{\delta E}{\delta W_i} = \frac{\delta E}{\delta y_n} \cdot \frac{\delta y_n}{\delta z_n} \cdot \frac{\delta z_n}{\delta W_i}
\]  

(2.7)

Recent deep learning models like autoencoder, denoising autoencoder and Deep Belief Network(DBN), can be learn better features but are trained greedily for a layer. These layers are then stacked one above the other to learn higher level features. Each layer learns a feature and the layer above it learns higher level of features from the features of the layer below. These algorithms are usually applied in unsupervised settings but can be modified for supervised settings.

### 2.2.6 Entity Linking

The task of linking textual entity mentions to entries in a KB that contain relevant information about the entities is called Entity Linking. Wikipedia (and its derivatives) are typically used as a KB in EL studies. Hence the task of automatically identifying concept mentions appearing in a document and linking it to a concept referent in a KB is referred as Wikification \([?\ ]\). The first works in EL appeared in 2006 \([?\ ]\) and 2007 \([?\ ]\). While Bunescu et al. tried linking named entities to Wikipedia, Mihalcea et al. \([?\ ]\) linked all concepts to Wikipedia. In 2008, Medelyan et al. \([?\ ]\) defined ‘commonness’ from ‘keyphraseness’. Commonness is the number of links with specific target as anchor text to total number of links with that anchor text. The concept was further refined by Milne and Witten as the Wikipedia Link based Measure (WLM) \([?\ ]\), which used only the hyperlink structure of Wikipedia to calculate semantic relatedness between any two words. WLM achieved a correlation of 0.68 with human and became the de-facto standard of semantic relatedness used in EL system. Many Wikification systems emerged ever since including the one using metadata of the document \([?\ ]\) and enhanced context using category tag \([?\ ]\)

### 2.2.7 Entity Linking in Short Texts and Tweets

With emergence and increasing popularity of short text and tweet, EL system designers are left with minimal contextual evidence to disambiguate the entities, unlike in the full document case. This shifted
the focus again to statistical measures like commonness and WLM. The TAGME [?] system uses both these measures to detect and disambiguate entities in short text and tweets. The system proposed by Meij et al. [?] identifies the concept of a tweet by linking the whole tweet. In other words, the system identifies the ranked list of entities a tweet can be associated to, semantically.

Entity linking in tweet Liu et al.[?] propose an Entity Linking (EL) system for tweets based on mention-mention similarity, mention-entity similarity and entity-entity similarity. Meij et al. [?] propose a machine learning model for disambiguating entities in tweets. This is the most exhaustive study for entity linking systems on tweets. Shen et al. [?] approach tweet entity linking problem by modeling user interests based on the assumption that each user has an underlying topic interest distribution over various NEs. Shen et al. integrate the intra-tweet local information with inter-tweet user interest information creating a graph based framework to link the NEs. Kosmix [?] is an industrial system that extracts, links, classifies and tags social media data, especially tweets. Whereas in our work, we focus on SNE identification, demonstrated on a filtering task. The Named Entity Linking in tweets is studied by Ibrahim et al. [?], who use Mention Normalization, Contextual Enrichment, and Temporal Entity Importance to improve it. They achieve 13% improvement in precision over AIDA [?] system, which is the state-of-the-art NEL system. EL system built by Jin et al. [?] concentrate on linking the entities that have limited web presence, often referred to as “tail entities”. They use a KB of 100M unique people from a popular social networking site, to demonstrate people-linking at the tail.

2.2.8 Knowledgebase Enhancement and Entity Linking

Most EL systems link mentions collectively, using coherence between entities [? ]. We studied coherence measures and datasets used in six recent[?]EL systems [? ? ? ? ? ?]. We see that CoNLL [?] and TAC2010 [?] are two popular datasets used for evaluating EL [?] on documents. There are three prominent coherence measures used in EL systems: (1) WLM, (2) Entity Embedding Distance and (3) Jaccard Similarity [? ? ]. WLM is widely acknowledged as state-of-the-art [? ]. Almost all the six approaches analyzed above, use WLM or its variant. Entity embedding Distance [?] is reported to give highest EL performance and is the baseline of our approach to tail entity linking (ELDEN). Both Yamada et al. and Barrena et al. [?] generate entity embeddings using word2vec model. However Barrena et al. use embeddings only to find similar words, and show improved EL performance using co-occurrence statistics from a large web corpora. Thus we adopted Yamada et al.s use of entity embedding distance in EL and improved it with web corpus co-occurrence statistics.

Recent research has looked at the problem of tail EL [? ? ]. Taneva and Weikum [?] suggest harvesting key-phrases from context for linking tail entities. ELDEN uses mentions of entities in context for linking tail entities. Mining additional context for linking tail entities was proposed by Li et al. [? ]. Hegde and Talukdar [?] overcome sparsity of edges by entity centric addition of new entities and relations. Li et al. [?] consider number of edges of entity, to judge the ease of doing EL on the entity, using edge

2 [?] presents a survey of EL systems.
3 Named Entity Disambiguation (NED) and EL are synonymous terms in research [?]
measures. Majority of approaches we analyzed above use edge based features. When an entity has less number of edges or no edges, we cannot rely on edge based methods for EL [?]. In ELDEN, PMI helps to increase edges of entities, leading to more robust coherence scores and therefore enhanced EL for tail entities. ELDEN proposes densifying the edge graph of entities. Densifying edge graph is also studied as ‘link prediction’ in literature [?]. ELDEN densifies the KG by a simple technique of adding edges using corpus mentions of entity. Since it results in improved EL performance, we believe that more sophisticated densifying techniques could yield better results.

**Word co-occurrence measures**: Chaudhari *et al.* [?] survey several co-occurrence measures for word association including PMI [?], Jaccard [?] and Co-occurrence Significance Ratio (CSR). Damani [?] proves that considering corpus level significant co-occurrences, PMI is better than others. Budiu *et al.* [?] compare Latent Semantic Analysis (LSA), PMI and Generalized Latent Semantic Analysis (GLSA) and conclude that for large corpora like web corpus, PMI works best on word similarity tests. Hence we chose PMI to refine co-occurring entity mentions in web corpus. We count entity mentions as entities in web corpus as tail entities.

### 2.3 Conclusion

In this chapter we looked at definition of the Named Entity Extraction problem, explaining Entity Linking and Knowledge bases. We saw how these two tasks go hand-in-hand in execution. We reviewed related literature on the two tasks. Entity Linking tasks starts by analyzing the words and phrases identified as NE by a Named Entity Recognizer (NER), and figures out the salient NE and NE attribute from the given set of identified NEs. So our literature review start from the best and state-of-the-art NERs and NERs for tweet. Then we look at literature on salient NE and NE attribute identification, both of which are modeled as supervised sequence prediction problem. Hence we review literature on sequence prediction models in NE identification subsequently.

After detecting the NE mentions, the mention are linked to KB. Hence related work on Entity Linking and Entity Linking on short texts and tweets follow. We complete the discussion with a review of methods on KB enhancement and methods on EL leading to KB enhancement, that is new entity identification for KBs.
Chapter 3

Overview of Approach

In this chapter, we explain our approach to Named Entity Extraction and Knowledge Base enhancement in a nutshell. Named Entities (NE) are extracted from text by linking mentions of NEs in text to a Knowledgebase (KB). This process is called Entity Linking (EL) \[ ? ? ? \]. It consists of two steps (i) identifying mention of NE in the text and (ii) identifying the KB entity of the mention. The KB stores facts about the NE like name, type, other such attributes and relationship with other entities in the KB. EL systems use these KB facts, to improve the EL task. Improved EL leads to more refined facts about entity getting stored in KB. Hence this thesis presents experiments on enhancing KB leading to improved EL and experiments on improved EL leading to predicting new entities to be added to KB. In this chapter, we look at EL improvement, how the two steps of EL (identifying NE mentions and identifying KB entities) are approached to achieve the improved results. We then look at how enrichment to KB is performed leading to better EL performance.

3.1 Identifying the Named Entity to link to Knowledgebase

Named Entity Extraction starts with analyzing the textual content with a Named Entity Recognizer (NER) module to identify mentions of NE in the text. State-of-the-art NERs identifies nearly all the NE mentions in the text. The NE extractor analyzes each mention by its importance and its relation to other NEs in the document. Identifying an NE by its importance or predicting the salience of an entity in a document, is a non trivial task. We refine this prediction by extracting further information about the NE specifically how the NE is related to other NEs in the document. Predicting if the NE is as an attribute of other NEs, is a related and equally challenging task which we solve in this thesis.

**Identifying the Salient NEs** A NE is salient when it is central to the document. In this thesis we propose a method to identify salient NE is documents. We present experiments on tweet as the document, assigning a salience score to each NE in a tweet. Salience score of a NE can be seen as the dominance score. In this sense the task of identifying Salient Named Entity (SNE) is equivalent to task of recognizing the dominant NE of a tweet.
Identifying the SNE is important in analysis of social media content. Social media content is a rich source of information and opinion, and its volume is growing by the day \(\text{[?]}\). However, social media posts are difficult to analyze since they are brief, unstructured and noisy. Many social media posts are about an entity or entities. Understanding which entity is central (Salient Entity) to a post, helps analyze the post better. Hence we propose a model that aids in such analysis by trying to identify the Salient entity in a social media post, tweets in particular. We present a supervised machine-learning model, to identify Salient Entity in a tweet and propose that the tweet is most likely about that particular entity.

We model SNE identification as a supervised sequence labeling task. In our approach, a supervised labeler learns ‘salience prediction’ from an annotated dataset. In devising this model, we have used the premise that, when an image accompanies a text, the text most likely is about the entity in that image \(\text{[?]}\). We build a dataset of tweets and salient entities and train our model on that dataset. The model itself is not dependent on tweets with images, since we use only text features of the tweet.

**Dataset and Results:** We created a dataset of 3646 tweets with SNE annotation. In our experiments we find that the model identifies SNE with an F measure of 0.63. We evaluate the model using a standard tweet-filtering task and our results are better than the median of the results. Our method outperforms two of the three baseline methods for salience identification. We have made the human annotated dataset and the source code of this model publicly available to the research community.

**Identifying NE as an attribute of another NE** We discussed about a method to identify the SNE of a document. We now look at how the identified NE is related to other NEs, i.e. identifying the NE as an attribute of another NE. Identifying the attribute relation or semantic category of a NE is an interesting problem. Recent efforts identify semantic categories like companies, products, movies, brands and more. We base our experiments on identifying a NE as semantic category of another NE, on product entities and specifically on product titles.

A large number of web queries are related to product entities. Studying evolution of product entities can help analysts understand the change in particular attribute values for these products over time. However, studying the evolution of a product requires us to be able to link various versions of a product together in a temporal order. While it is easy to temporally link recent versions of products in a few domains manually, solving the problem in general is challenging. The ability to temporally order and link various versions of a single product can also improve product search engines. In this study, we tackle the problem of finding the previous version (predecessor) of a product entity. Given a repository of product entities, we first parse the product names using a CRF model. After identifying entities corresponding to a single product, we solve the problem of finding the previous version of any given version of the product. For the second task, we leverage innovative features with a Naïve Bayes classifier.

Our approach is to identify the category of a NE using the fact that parts of the name of a NE contain information about its category. This is especially true of entities like product names. For example in the NE “*Canon IXUS 160*”, word ‘*Canon*’ indicates the brand, word ‘*IXUS*’ indicates the product and ‘*160*’ indicates version. In the proposed approach, we parse the product title to predict each word as belonging to the category brand, product, version or other (none). This is a tag prediction task. It draws
evidence from the features of the neighboring tags. Hence we model it as a sequence labeling problem. We propose to solve it by training a sequence labeler on a human annotated dataset.

Figure 3.1: Two stages of NE attribute identification.

We solve the problem of finding the predecessor version of a given product version in two stages, shown in Figure 3.1. In the figure oval boundary indicates brand, rectangular boundary indicates product and triangular boundary indicates version. In the first stage, given a set of product entity names, we parse the product names to identify the “brand”, “product”, and “version” indicating words from the product name. The first stage thus gives us all the product entities belonging to the same product as the given (query) product. The second stage ranks these candidates and chooses the most probable predecessor version from the candidate set. Both the stages follow a supervised approach with interesting features.

Dataset and Results: Experiments on a dataset crawled from Amazon show that our methods achieve a precision of \( \sim 88\% \) for the first stage, and a precision of \( \sim 53\% \) for the second stage.

### 3.2 Linking Named Entities to Knowledgebase

The KB stores the canonicalized name of NE as entity. For example the phrases “Obama” and “President Obama” link to “Donald_Trump”, which is the canonicalized name. While a NER identifies the NE phrases “Obama” and “President Obama”, linking it to “Donald_Trump” helps resolve synonymy. A NE can represent many entities in KB. For example, the NE “Stanford” may refer to “Stanford University” or the place “Stanford”. This is the polysemy problem. Among the entity options in the KB, finding which entity the mention refers to, is called entity disambiguation. Entity disambiguation is performed by linking the NE mention to the corresponding KB entity.

Entity Linking consists of identifying mentions of NEs \((N)\) in a given text and linking them to KB entities. A NE mention \(m\) can refer to \(E\) different entities in the KB. Mention detection is the task of identifying the set of all mentions in a document \(M\) such that \(M = \{m_i, e_i\}_{i=1}^N\) where \(e_i = \{e_j\}_{j=1}^E\).
Entity disambiguation is the task of choosing the correct entity $e$ to link to $m$ among $E$ entity candidates. Linking decides whether the entity chosen in the disambiguation stage is the target entity for the NE $m$. When none of the $E$ candidates is the target entity, EL system links $m$ to a NIL entity. Linking to a NIL entity leads to new entity identification.

In the disambiguation of mention, textual context of the mention plays an important role. The challenge in disambiguation is that many NE strings rarely appear even in a large training corpus. Therefore the system must identify them only based on context. However the size of textual context has reduced with evolution of Web. Text size has reduced from documents (web-pages and blogs) to short-text (tweets and queries). This has led to increased emphasis on techniques to improve inference from the (lesser) available context and the use non-textual signal like (global) coherence of entities in the KB. Accordingly we have different approaches for linking the mention, for different context lengths (Figure 1.3).

**EL on Documents** The text of a document has a well defined grammatical structure in contrast to social media generated text. So we call the text of document as ‘structured’ text. The structure aids in prediction of NEs in the text with very high confidence.

**Dataset:** In our experiments on EL on documents, we use the dataset from Text Analysis Conference (TAC) Knowledge Base Population track. This dataset consists (train and test set) of EL query documents and a KB derived by automatically parsing Wikipedia, called TAC KB hereafter. The query document consists of entity linking annotations. The mentions are annotated with either gold standard entities in the KB or NIL, if there is no corresponding entity in the KB.

For linking mentions in a document, the EL system utilizes the associated document to provide context that is used in linking the mention. We approached the EL task as four sequential steps (Figure 5.1).

1. **Named Entity Recogniser (NER):** We use a Stanford NER module which uses surface and contextual evidence of text to label sequence of words in the text as NE. It labels the names of things such as person and company as NE mention and predicts the NE type for the mention (PERSON, LOCATION and ORGANIZATION).

2. **Named Entity Disambiguator (NED):** The NE mention identified by NER could be polysemous. We resolve polysemy using two indices created by processing Wikipedia. Wikipedia article text frequently contain hyper-links to other Wikipedia pages. The anchor text of the hyper-link to a Wikipedia page is a good human-edited representation (or explanation) of the Wikipedia page it points to. Many works in Entity Linking and linking to Wikipedia exploit this hyper-link structure of Wikipedia for disambiguation. In similar lines we extracted all hyper-links between Wikipedia articles (KB nodes) to create two indices, ‘Anchor Index’ and ‘In-Link Index’. ‘Anchor Index’ maps Anchor tag to $KB$-ids. This helps calculate the Link Probability which is the probability with which the given word (phrase) occurs as a link in Wikipedia. ‘In-Link Index’ maps $KB$-id to List of $KB$-ids, which is used in calculation of bias or Prior Probability. This is the probability that a given $KB$-id will link to another $KB$-id based on its previous occurrences. We also indexed the TAC KB using Lucene. On querying this index for a word or phrase, Lucene index provides all $KB$ nodes that are related to the
word and a score which showed relevance of the retrieved KB node for that word. We used the Lucene score as a feature to train the EDL training model. Further Lucene also indicates if the word (phrase) was found in the title/infobox/text/category of the KB entry, which is used as position rank feature. Using the indices the NED produces disambiguations \((d(\cdot))\) for each mention \((m)\), as a set of \(N\) possible entities \(e\). For each \(e\) the feature values \((F)\) namely Link probability, Prior probability, Lucene score and Position rank, corresponding to \(e\) is also computed.

\[
d(m) = [e,F]_{n=1}^{N}
\]

**III. Classifier:** We train a logistic regression classifier (SVM) to combine the four feature values as the confidence of predicting the entity for the given mention. For training, we used named entities identified from the TAC training document collection (LDC2014E15).

**IV. Linker:** All the predicted entities are ranked according to the confidence value and the entity with highest confidence is chosen. Since the data is highly biased towards the 0 (negative) class, we obtained very low confidence scores for 1 (positive) cases. If the confidence of prediction is below a threshold \(\tau\) or if no entity was predicted (all of \(e\) is classified as 0), the linker links the mention to NIL.

**Results:** In the development test dataset (LDC2014E54) our system obtained a Precision (P) of 0.788 and Recall (R) of 0.19. Our approach disambiguates Named Entity (NE) where only limited amount of entity information is present. It did not use any external information source.

**Entity Linking on Tweets** Social media networks like Twitter have emerged to be major platforms for sharing information in the form of short messages (tweets). Analysis of tweets can be useful for various applications like e-commerce, entertainment, recommendations, etc. Tweets are short text messages with a maximum length of 140 characters. Our approach disambiguates the NEs in the tweets based on three different measures: (1) Wikipedia’s context based measure (M1); (2) anchor text based measure (M2); and (3) Twitter popularity based measure (M3).

The NE mentions are detected using two Twitter part-of-speech (t-POS) taggers. It was observed that both of the t-POS taggers did not extract all proper noun sequences. In order to improve recall we merge the results from both. The tweet text is fed to the system and the longest continuous sequences of proper noun tokens detected using the above approach are extracted as the NEs from the given tweet. The NE mentions are disambiguated using the scores from the three measures, combined using LambdaMART model, which gives the final disambiguated entity. The measures are

**Wikipedia’s Context based Measure (M1)** This measure disambiguates a NE mention by calculating the frequency of occurrence of the NE mention in the Wikipedia corpus. Wikipedia’s context based measure has been used in various approaches for disambiguating NEs in tweets. We queried a Wikipedia index with the mention, to get the candidate entities in the ranked order. Each candidate entity is assigned its reciprocal rank as score.

**Anchor Text based Measure (M2)** Google Cross-Wiki Dictionary (GCD) is a string to concept mapping, created using anchor text from various web pages. A concept is an individual Wikipedia article, identified by its URL. The text strings constitute the anchor hyper-texts that refer to these concepts.

\(^1\)We chose \(\tau = 0.25\) after observation
Thus, anchor text strings represent a concept. We query the GCD with a NE mention along with the tweet text. Based on the similarity to the query string, a ranked list of probable candidate entities are created (which is the ranked list using M2). The ranking criteria is Jaccard similarity between the anchor text and the query. So if the NE is highly similar to the anchor text, then the corresponding concept will have a high score.

**Twitter Popularity based Measure (M3)** Tweets about entities follow a bursty pattern. Bursty patterns are the bursts of tweets that appear after an event relating to an entity happens. We exploited this fact and tried to measure the number of times the given NE mention refers to a particular entity on Twitter recently. The NE mention is queried on Twitter API and the resultant tweets are analyzed. All the tweets along with the NE mention are then queried on the GCD and the candidate entities are taken. Based on the scores returned using GCD, all the candidate entities are ranked (which is the ranked list using M3). As Twitter popularity based measure captures the people’s interests at a particular time, it works well for entity disambiguation on recent tweets.

In essence, the methods M2 and M3 are similar but with different inputs. Both use GCD, and produce candidate NEs and score as output. However, M2 takes NE mention and single tweet text as input whereas M3 takes NE mention and multiple tweets as input.

**Dataset and Results:** The dataset comprises of 2300 tweets annotated with the entity mention and its corresponding DBpedia URL. Our approach achieved an F1 of 0.512 on the test set. The best results are obtained when combination of all the measures are used for disambiguation.

**Entity Linking on Queries** Our approach for EL on search queries derives motivation from the work by Ferragina et al. [?] for linking entities over short documents. Ferragina et al. explored Wikipedia in-links for detecting and linking the entities present in search queries. The NEs are detected based on the probability that the NE appears as an anchor link on Wikipedia. The disambiguation is done based on the probability of the NE linking to a particular KB entity and a Wikipedia-based semantic relatedness measure as proposed by Milne et al. [?]. The set of links chosen are pruned based on the coherence among the entities detected within the text document.

We used the following Wikipedia-based measures: Wikipedia Link-based Measure (WLM) [?], Senses of Anchor \( Pg(a) \), Prior Probability \( Pr(p|a) \) and Link Probability \( lp(a) \). We used ‘Anchor Index’ and ‘In-Link Index’ mentioned earlier in this section. NE mentions are detected using link probability. Unlike documents and tweets that we discussed so far in this section, queries have much lower context length. This rules out use of NERs to arrive at NE mentions. Hence for detecting NE mentions, we take continuous word sequences of up to 6 words and find if the string appears as an anchor in Wikipedia. If the probability of being an anchor is greater than a predefined threshold, then the anchor is taken as a detected NE and all the pages referred by it are taken as possible candidates for NE disambiguation.

---

2[https://dev.twitter.com/docs/api/1.1/get/search/tweets](https://dev.twitter.com/docs/api/1.1/get/search/tweets)

3Unique pages in Wikipedia, an anchor links to
However this involves a large number of look-ups on the Wikipedia anchor index. In order to reduce the number of look-ups, we propose using mention filtering including ‘Stop-word Filtering’ and ‘Twitter POS Filtering’. We adapt the disambiguation function from the TAGME system, after making a few changes for efficient computations. The overall score given to a candidate $p_a$ consists of the relatedness score $rel_a(p_a)$ and the prior probability $Pr(p_a|a)$ of the candidate. These two factors are combined to get the overall score. Finally we also prune the annotations suggested, to possibly discard the meaningless annotations. This pruning is done using a scoring function. Pruning score $\rho$ combines the link probability and coherence. The NE mentions with $\rho$ value less than threshold are pruned.

Dataset and Results:

For the ERD Challenge, used snapshot of Freebase as KB. This contains only those entities that have English Wikipedia pages associated with them. The dataset consisted of web queries annotated with the entities in this KB. As the evaluation of the ERD challenge was restricted to these entities, we indexed this dataset and restricted our final results to these entities. The test set consists of 500 search query strings. For each query, the system outputs the entity mentions and the Freebase ID from the given KB, that the mention links to. Our system achieved an F1 of 0.53. While we reached sixth place in terms of results, we were first in terms of achieving lowest latency in the challenge.

3.3 Knowledgebase Enhancement and Entity Linking

Knowledgebase Enhancement for Entity Linking

EL systems primarily exploit two types of information: (1) degree of similarity between the mention and the candidate entity string, and (2) coherence measure between the candidate entity and other entities mentioned in the vicinity of the mention in text. Coherence measure essentially captures how well the candidate entity is connected, either directly or indirectly, with other KB (Wikipedia) entities mentioned in the vicinity. State-of-the-art EL system measures coherence as distance between embeddings of entities. Yamada et al.’s EL system performs well on entities that are well connected in Knowledge Graph (KG) and are popular, but not so well on less popular entities which are sparsely connected in KG. We present a method to improve tail entity linking performance using KG densification. Edges to tail entity in the KG is increased by adding edges from mentions of entities co-occurring with it in web corpus.

As a first step, we analyze how the degree of a knowledge graph entity affects its EL performance. Consider the sentence shown in text in Figure 3.2. There are two mentions (underlined) in the sentence. The figure also shows linkable candidate entities in KG for these mentions. Using a conventional EL system, the first mention Andrea Broder can be linked to Andrea Broder using string similarity between the mention and candidate entity strings. This is unambiguous. Second mention WWW has two candidates, World Wide Web and WWW conference, and hence ambiguous. In such cases,
coherence measure between the candidate entity and other unambiguously linked entity(ies) is used for disambiguation.

State-of-the-art coherence measure, Wikipedia Link based Measure \[? \] (WLM) uses edges to measure entity coherence. WLM measures coherence using the number of common edges i.e. edges World_Wide_Web shares with Andrei_Broder, as compared to edges WWW_conference shares with Andrei_Broder. But WWW_conference has less number of edges (it is a tail entity) compared to World_Wide_Web. This leads to poor performance by WLM method in this disambiguation resulting in WWW not linking to WWW_conference. Yamada et al.’s method uses WLM for training entity embeddings, leading to poor performance on linking tail entities. Thus number of edges of candidate entity affects EL performance.

We proposed ELDEN, an EL system that addresses this edge sparsity problem of tail entities. ELDEN uses information about entities available in the web to solve the problem. It uses a web corpus to find co-occurring entity mentions and refines the co-occurrences with Pointwise Mutual Information (PMI) \[? \] measure. It then adds edges to the entity, from mentions of entities that have a positive PMI value for the given entity. In Figure 3.2, mention of entity Personalization co-occurs with mentions of Andrei_Broder and WWW_conference in web corpus and has a positive PMI value with both. So ELDEN adds edges from Personalization to Andrei_Broder and WWW_conference. This densifies neighborhood for entities. Coherence measured as distance between entity embedding where embeddings are trained on this densified entity neighborhood, results in improved EL performance.

Now we analyze why degree of knowledge graph entities affect EL performance \[? \]. Figure B.1 of Appendix B.2 shows degree (edge count) vs. number of entities having that degree in the KG. We see that few entities have many edges and many entities have few edges. Entities that have many edges are called head entities and entities that have less number of edges are called tail entities. The number of tail entities in a KB is many folds larger than the number of head entities \[? \]. In Figure B.1 we

---

7This section 3.3 focuses on mention disambiguation. We assume mention and candidate entities are detected already.
8Analysis of edge-count and number of KG entities with that edge count is presented in Appendix B.2
find that entities with 500 edges or less make up more than 90% of Wikipedia. Also state that the main difficulty in EL is in linking tail-entities. Thus linking tail entities accurately is critical to EL performance.

**Dataset and Results**: ELDEN is evaluated on CoNLL and TAC2010 are two popular and benchmark datasets used in recent years for evaluating EL on documents. ELDEN performed on par with state-of-the-art EL methods while it is most effective in case of tail entities. It presents the first ever comparison of performance of state-of-the-art EL systems in linking tail entities. ELDEN source code will be made public. ELDEN uses mentions of entities from a web corpus refined with PMI measure, to improve EL performance. As it uses a corpus based measure (PMI), it is general purpose and may be applied to improve EL involving any KB.

**Entity Linking for Knowledgebase Enhancement** In our day-to-day life, we encounter new and interesting entities (e.g., a person’s name or a geographic location) while reading text. New Entity Identification (NEI) is the process of automatically identifying an entity present in text, but not present in a Knowledge Base (KB). Understanding NEI approaches is critical in the automatic construction and maintenance of KBs.

Our approach starts with reviewing the literature on NEI approaches of Entity Linking (EL) systems. We examine recent findings, best-result algorithms, and state-of-the-art systems pertinent to NEI research, while assessing the reproducibility of the results. We identify two prominent clusters of NEI approaches. Then, we re-implement approaches from both the clusters, and make several observations from our experiments. Our findings answer the following questions: How EL features impact NEI; will the use of a dedicated classifier for new entities improve NEI performance; and finally, how the standard EL systems can be improved to achieve better NEI performance.
Chapter 4

Identifying the Named Entities to Link

Analysis of textual content proceeds with the aim of identifying the important information contained in the text. Importance of information as instilled in the NEs dealt with in the text, is extracted in Named Entity Extraction. The text under analysis typically deals with multiple NEs. Hence understanding the importance of one NE with respect to other NEs in the text, is crucial to success of Named Entity Extraction. In this thesis we propose two methods to assess the importance of NE in the text. First method predicts ‘salient’ NEs of the document from the NE’s syntactic and semantic structure. Second method predicts importance of NE by its relation to other NEs in the text. These methods are elaborated in this chapter.

We base our study of ‘identifying the salient NE among the NEs in a document’ on ‘identifying salient NEs in tweet’. We choose social media data and tweet data in particular for the following reasons. Social media platforms Twitter is fast emerging as an important forum of public opinion, starting from governments and celebrities to common man using it as the medium to listen to and disseminate opinions. With increasing popularity the amount (volume) of data (tweets) has also increased.

Extracting information from tweets is challenging as tweets have reduced context (maximum 140 characters), are dynamic in nature and are less grammatical than longer posts. Recent studies indicate that most tweets have multiple entities. This necessitates a method to determine the prominence—the salience—of entities in these tweets. Thus the challenge of estimating entity salience is unique, inherently difficult and thus far un-addressed in research. It can also be stated as identifying the right NE to extract from a tweet or figuring out which among the many NEs, does the tweet talk about. This problem and the solution presented in this thesis holds good for different types of content like web-page content, user search query and microblog posts.

We saw in Chapter 2 that research in Named Entity Recognition (NER) and thence Named Entity Extraction (NEE) is about two decades old. Various supervised methods have been proposed and the current state-of-the-art is Conditional Random Fields (CRF). In using CRF, the NER is modeled as a sequence labeling task. Words are classified into one of the categories (i.e. tagged with one of the tags)

1 On an average, there are 6000 tweets posted each second as per http://www.internetlivestats.com/twitter-statistics/
2 In their very extensive study of evolution of Twitter ecosystem, Liu et al. say that ~50% percentage of tweets contain entities and average number of entities per tweet is 1.3.
with a tagging scheme. The industry standard tagging scheme is BIO tagging scheme. BIO tagging scheme subdivides the tags as begin-of-entity (B-) or continuation-of-entity (I-) and Non entity (O-). CRF tagging uses the conditional probability of current and previous words based on features. CRFs have been quite successful and we propose to use it in our experiments in this chapter which includes predicting the salient NE of a tweet and identifying the semantic category (brand, product and version names) from a product title.

4.1 Salience of a Named Entity

What is salience?
Salience can be understood as the aboutness of a document, as in what is the document talking about? So salient entity becomes the important entity talked about in the document. Delort [?] defines salience of an object as ‘its capacity to attract or catch an observers attention’. As a concept Salience has been defined as aboutness, most noticeable, conspicuous and prominence in dictionaries. The fact that so many words have been used to define a concept tells the difficulty in defining the concept in practice. Gamon et al. [?] make two interesting observations on this definition.
(1) Salience is a function of the structure of text, and indirectly a function of the intention of the author, as opposed to a function of information need. Boguraev & Kennedy (6) in their work resonate the later view when they say “Salience is a measure of the relative prominence of objects in discourse: objects with high salience are the focus of attention; those with low salience are at the periphery”. Thus we can see that in our context of set of named entities in a tweet, a Salient Named Entity (SNE) is the NE that is central to the tweet.
(2) Salience is not same as relevance. Relevance is a well-understood in the IR community. Given a document and an information need, if the contents of the document satisfy the information need, then the document is relevant to the information need or query. Thus relevance is a function of document and query, as in the same document can be relevant to one query and not relevant to another. Whereas salience is a function of document contents. Hence it remains constant for a document.

Why salience?
Identifying the salient entity of a tweet can be difficult when done by humans also, due to the differing perspectives of salience. For example, consider the tweet “Google Executive Dan Friedenburg Dies in Everest Avalanche Nepal Earthquake #google #techtalent sorry for your loss”. Annotator A1, who is a technology enthusiast, marks “Dan Fredinburg” and “Google” as SNEs. Annotator A2, interested in mountaineering, marks “Everest” as SNE. Annotator A3, interested in current affairs, marks “Dan Fredinburg” and “Nepal” as SNEs. We see that SNE identification varies with perspective and/or interest of the annotator. This variance is captured as relevance. The variance can lead to bias in applications like reputation management and newsmaker identification. In these applications, an analyst or a program is monitoring tweets to assess the impact of an entity. Here tweets need to judged based on whether the entity of interest is salient to the tweet (central to the tweet...
and hence is intended by the author of the tweet), rather than whether the analyst found it important. Biased judgment based on relevance perceived by analyst can lead to incorrect judgment of reputation impact calculation and newsmaker identification. Hence we need to identify salient entity in these cases.

**How to identify salient entity?**

In our approach, a supervised labeler learns ‘salience’ from an annotated dataset. The dataset consists of 10,938 tweets manually annotated with the salient named entity and the Wikipedia page that describes it. When author (originator of the tweet) includes an image in the tweet, the image captures the salient entity. This is the key intuition in deciding the SNE in our proposed approach. This assumption was proven by Deschacht *et al.* who analyzed cross-media entity recognition in nearly parallel visual and textual documents. Deschacht *et al.* prove that there is higher probability that salient entity is present in the image and the ratio of entities in the text that are present in the image is 22.96%. Of late tweets containing images are increasing in number and Twitter is encouraging inclusion of images in tweets. Hence in our proposed approach, we take the entity that is present in the image accompanying the tweet as the salient entity of the tweet.

### 4.2 Approach to identifying salience of Named Entities

We present a supervised learner based solution for the problem of identifying the salient NE of a tweet. The task of selecting the NE that is salient to the tweet is best done by human intelligence. Hence, we create a human-annotated dataset consisting of tweets and thier salient NEs for this task and train a learner on it. To start with, we identify the NEs in a tweet by using a set of three NERs. These NEs are candidate SNEs for a tweet. To select the SNE from the candidate SNEs, we train a learner. The learner model associates a salience score for each candidate SNE, which is the probability of a NE being SNE. As we model this task as a supervised sequence labeling task. The sequence labeler is trained using on the salience dataset.

---

4.3 Experiments on identifying salience of Named Entities

4.3.1 Dataset

‘Salience’ is learned by the sequence labeler by training on a salience dataset consisting of tweets annotated with its SNE. However annotating a tweet with its SNE is a difficult task owing to annotator bias as described earlier. This is overcome using the image accompanying the tweet as an evidence of salience. As the image conveyed the author’s intended entity, our annotation guidelines (See Appendix A for annotation guidelines) instructs the annotators to select the NE as the SNE only if it is present in the image. Thus we are able to address the issue of annotator bias (i.e reduce the variance) in the choice of SNE.

The annotators background knowledge and interest in the domain of the (tweet) text was important in the SNE identification task. The task was selecting the salient of the NEs required the annotators to recognize all the NEs in the tweet. Further the annotator also need to know how the entity looks in order to identify the SNE in the image. With these requirements we were constrained to pick only the domains that were popular among the annotators such that annotators had good knowledge of it. Considering our annotator’s interest we chose the popular sporting event (cricket world cup), a popular entertainment event (annual film award) and a much awaited product release (apple watch release). Considering the age group of our annotators, the popularity of these events ensured annotator’s domain knowledge and hence high quality of the annotation.

Existing Datasets The RepLab task\footnote{Evaluation exercise for Online Reputation Management systems. http://nlp.uned.es/replab2013/} tries to analyze tweets for potential mention of entities, filtering those tweets that refer to the entity. We use this task and dataset to evaluate our system. We also use the dataset created by Ghosh et al.\footnote{collected by crawling http://twitter-app.mpi-sws.org/whom-to-follow/} to identify the expert twitter users for a topic. The dataset provided by Meij et al.\footnote{http://nlp.uned.es/replab2013/} contains both tweet and the corresponding Wikipedia article, which are the concepts of these tweets. Though this originally contained 502 tweets, we could get only 363 tweets owing to twitter accounts getting banned and tweet deletions. As this size was not sufficient for creating a learner and as we are interested in annotating tweet with only SNEs (not all entities or NEs), we annotated tweets to create the CWC15 dataset.

CWC15 Dataset We created the dataset of tweets annotated with their SNE, which we call CWC15 dataset hereafter. Using the presence of NE in the image accompanying the tweet as evidence of salience, we annotate the NE as SNE of the tweet. Here we use only those tweets that have an image included in it, in constructing the CWC15 dataset. We use the textual content from tweet for SNE Identification. The image is used only for the purpose of helping the annotators decide the SNE. No other feature of the image is used in the labeler, similar to the approach of Deschacht et al.\footnote{http://nlp.uned.es/replab2013/}. In other words, the restriction that tweet should contain an image is applicable only to the tweets for CWC15 dataset creation and not for the tweets for which SNEIT identifies the SNE.
Preprocessing

The tweets for CWC15 dataset creation were collected during the ICC Cricket World Cup 2015[6]. Hashtags of the quarter final matches have been used as queries. We collected tweets that contain at-least one image (discussed in Section 4.3.1), is in English language and is not a re-tweet. Total corpus size is 10,938 tweets.

![Annotation Interface](image)

**Figure 4.2:** Annotation Interface

**Manual Annotation** In order to obtain manual annotations (both for training and evaluation), initially we asked six volunteers to manually annotate upto 2000 tweets each. The volunteers were graduate students in the age group of 20 to 30 with interest in the game of cricket. The group had three male and three female members. The annotators were presented with an annotation interface shown in Figure 4.2. The NEs in the tweet identified by the three NERs (discussed in Section 4.3.2) were presented. The annotation guidelines specified that the annotator should choose an NE being salient to the tweet only if it is present in the picture. On choosing an NE, the Wikipedia entities of the chosen NE were presented.

and the annotator chose the Wikipedia page that best describes the SNE of the tweet. They could also choose more than one NE in cases where multiple SNEs exist. The annotator was asked to mark the tweet as D for Duplicate (Tweet text and/or image repeats in the dataset) or P for Pointless (Pointless conversation or Image not containing any NE or Image containing non English text or Advertisement) or S for Sarcastic (tweet text is sarcastic with respect to image) or N for Not Annotatable (NE is not presented or invalid) when the respective conditions were satisfied.

Inter Annotator Agreement on Methods
The annotators are able to agree on the salient entity of a tweet when the author includes an image in the tweet. In order to measure the reduction in variance due to the presence of image, we created a smaller dataset by randomly sampling fifty tweets from CWC15 dataset. We asked four annotators to annotate this smaller dataset. The annotators were first presented with the 50 tweets without their images. After the first 50 tweets are annotated, they were presented with the same 50 tweets, with the images. The inter annotator agreement is measured using Agreement percentage (AP) and Cohen kappa [?] as presented in columns ‘With Image’ and ‘Without Image’ of Table 4.1. Agreement percentage is the percentage of times when both the annotators have agreed on same set of SNEs (and KB entries). Cohen kappa is better than agreement percentage since it takes into account the agreement occurring by chance.

<table>
<thead>
<tr>
<th>Measure</th>
<th>With Image</th>
<th>Without Image</th>
<th>Overall</th>
<th>Cricket</th>
<th>Movie</th>
<th>Product</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agreement SNE</td>
<td>68(+17%)</td>
<td>58</td>
<td>66.66</td>
<td>70</td>
<td>70</td>
<td>60</td>
</tr>
<tr>
<td>Percentage SNE and KB</td>
<td>58(+16%)</td>
<td>50</td>
<td>58.33</td>
<td>60</td>
<td>60</td>
<td>55</td>
</tr>
<tr>
<td>Cohen Kappa SNE</td>
<td>67(+28%)</td>
<td>52</td>
<td>72.85</td>
<td>74.76</td>
<td>74.57</td>
<td>64.61</td>
</tr>
<tr>
<td>Cohen Kappa SNE and KB</td>
<td>53(+51%)</td>
<td>35</td>
<td>60.25</td>
<td>62.01</td>
<td>62.29</td>
<td>50.73</td>
</tr>
</tbody>
</table>

Table 4.1: Inter annotator agreement scores

The annotation method that use the image has a better agreement score of about 67% when compared against the method without the image. The higher inter annotator agreement shows better agreement among annotators when the SNE evidence is provided in the image. This result has motivated us to annotate every tweet in CWC15 dataset by only one annotator, thereby quickening the annotation process. One human annotator annotated SNE across text-image pairs in the experiments of Deschacht et al. [?] too.

Inter Annotator Agreement on Domain
In order to measure the variation of inter-annotator agreement over domains, we created a smaller dataset with tweets from three different domains ‘Product’, ‘Cricket’ and ‘Movie’ using keywords AppleWatch, SAvsNZ and NationalAwards respectively. We randomly sampled 20 tweets satisfying the tweet selection conditions discussed in Section 4.3.1 from the three domains and asked two annotators to annotate the 60 tweet corpus. The inter annotator agreement
measured using Agreement percentage and Cohen Kappa (as in Section 4.3.1) is presented in Table 4.1, columns ‘Overall’, ‘Cricket’, ‘Movie’ and ‘Product’. Agreement percentage for both SNE and KB entry is good for cricket and movie domains. The Cohen Kappa scores show ‘fair agreement’ [?] and is comparatively higher for cricket domain in SNE annotation. Thus we chose cricket as domain.

CWC15 Dataset Statistics A total of 10,938 tweets were annotated by volunteers to create the CWC15 dataset. Out of these, 4272 were marked duplicate, 507 were marked as sarcastic and 1838 were marked as pointless. For 307 tweets, no NEs were identified by NERs and for 368 tweets, no KB entry was identified. After discounting these, we are left with 3646 tweets. These 3646 tweets have an NE marked as salient and for a subset of 1812 tweets, a Wikipedia title representing that SNE is chosen. In other words, a salient NE exists only for 33% of the tweets and a Wikipedia entity is identified only for 16.57% of them. We use the 3646 tweets as the positive samples for the labeler and 1812 tweets for training the tweet linker in the experiments and results that ensue in this article. Out of the 10,938 tweets, 8425 have an NE and 3646 have an SNE, which put the probability of an NE becoming an SNE in CWC15 dataset at 0.43. Figure 4.3 presents an analysis of the types of SNEs that are identified by the annotators. Half of SNEs are of the type persons, 27% are locations, 11% are organizations and 2% are events. Others including ‘none’ made up 9.5%. Thus SNE annotations of CWC15 dataset are spread across the entity types while the SNE annotations of comparable systems [?] are limited to names.

Why Cricket World Cup The Cricket World Cup 2015 (CWC2015) held in Australia and New Zealand created unprecedented levels of online and social media interest, as can be seen from the official statistics of the CWC 2015 web page of the event organizing body. There were over 26 million unique visitors to the official cricket world cup website during the first 30 days of the 45 day event. These visitors made up over 225 million Page views. The video clips of matches of the tournament was watched by fans from over 200 countries with over 24 million video plays combined across website and App. On Social Media there was more interest than ever seen before for a global cricket event as people in millions interacted with the tournament. On Facebook, 36 million people generated 341 million interactions, with

---

cricket becoming a regularly trending topic throughout February and March. On Twitter, the discussion around #cwc15 was huge, with over 8 million tweets sent around the tournament, with over 800 Million live tweet impressions from the group stages. In comparison, the 2012 London Olympics generated 150 million tweets. Thus the CWC 2015 was one of the most engaging topics on social media and twitter, producing significant amounts of data convincing us to create a dataset around this topic.

4.3.2 Experiments

The first step in analyzing tweets as in most IE pipelines, is the identification of NEs using Named Entities Recognizer for tweets [? ? ? ]. A NER may identify none, one, or more than one NEs in a tweet. Among the NEs, identifying the SNE is a novel and non-trivial task, which we address in this article. We identify the NEs in a tweet by using a set of three NERs. These NEs are candidate SNEs for a tweet. To select the SNE from the candidate SNEs, we train a learner. The learner model associates a salience score which is the probability of a NE being SNE, for each candidate SNE. We model this task as a supervised sequence labeling task. The sequence labeler is trained using a robust set of features. The features include POS tag (as in NNS, VBP, PRP and so on), Chunk POS tag (like BNP, I-NP and so on) and Entity tag (B-ENTITY and I-ENTITY) extracted from NER.

We call our system of ‘Salient Named Entities Identification in Tweets’ as SNEIT. The architecture of SNEIT system is shown in Figure 4.4. For an incoming tweet, the NERs identify the candidate SNEs. SNEs are selected from the candidate SNEs using a supervised sequence labeler.

![Figure 4.4: SNEIT System Architecture](image)

**Candidate SNE Generation** NE identification task poses many challenges, especially for tweets [? ]. In rare cases the NEs identified are either few or are completely incorrect. To overcome these and ensure better recall of NEs, Bansal et al. [? ] combine the outputs from different NER systems. This heuristic has been successfully used by the proposed work. For identifying NEs, we propose to use three NER systems, which are reported to give better F1 scores by Derczynski et al. [? ]. They include Ritter et al. [? ], Gimpel et al. [? ] and Finkel et al. [? ]. All of them are open source NERs and are used ‘as is’ in the SNEIT system. Their results are merged (union of subset) to obtain the candidate SNEs, $SNE_{Candidate}$.

**SNE Identification** The task of identifying SNE is modeled as a sequence labeling problem. This is done using Conditional Random Fields (CRF) [?], a popular machine learning algorithm that assigns tags to token sequences [? ]. CRF considers the features of current and neighboring tokens for tag assignment [?]. In our implementation, we propose to use 15 features of a token. They are based on POS

---


10 We used the implementation [https://github.com/tpeng/python-crfsuite](https://github.com/tpeng/python-crfsuite)

11 This puts the window size as 3. We experimented with window size of 5 too. Window size of 3 gave best results which we reported here.
tag, Chunk POS tag and Entity tag, extracted from NER. We are interested in the SNE tag. We used the industry standard BIO encoding. It subdivides the tags as begin-of-entity (B-) or continuation-of-entity (I-) and Non entity (O-). Thus our target labels are B-SNE, I-SNE and O-SNE.

Twitter NLP toolkit\[12\] is used to tokenize the tweets and extract features of tokens. These features are shown in Table 4.2.

<table>
<thead>
<tr>
<th>Feature Type</th>
<th>Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>POS tag</td>
<td>NNS, VBP, PRP</td>
</tr>
<tr>
<td>Chunk POS tag</td>
<td>B-NP, I-NP, B-VP, I-VP</td>
</tr>
<tr>
<td>Entity tag</td>
<td>B-ENTITY, I-ENTITY</td>
</tr>
</tbody>
</table>

Table 4.2: Features of labeler.

The CRF labeler encodes tokens with tags B-SNE, I-SNE and O-SNE, which are sufficient to identify the set of tokens contributing to a SNE. The features used by the CRF labeler along with their respective weights are shown in Table 4.3. Labeler was trained using tweets of CWC15 dataset (explained in Section 4.3.1).

### 4.3.3 Results

In this section we present the SNE identification performance of SNEIT, where we compare and discuss the design options considered.

<table>
<thead>
<tr>
<th>NER</th>
<th>Average NEs per tweet</th>
<th>SNE tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ritter et al.</td>
<td>2.22</td>
<td>B-SNE 0.74</td>
</tr>
<tr>
<td>Gimpel et al.</td>
<td>2.21</td>
<td>I-SNE 0.59</td>
</tr>
<tr>
<td>Finkel et al.</td>
<td>0.89</td>
<td>O-SNE 0.93</td>
</tr>
<tr>
<td>Combined</td>
<td>3.76</td>
<td>Overall 0.90</td>
</tr>
</tbody>
</table>

Table 4.4: NER Comparison

Table 4.5: SNE Identification

Candidate SNE Generation NERs are used to identify the NE mentions in the tweet. This has two possible outcomes. In first case, the NER does not identify any NE in the tweet. We consider this tweet as a tweet for which SNE does not exist or cannot be determined (These tweets are marked P and N during annotation). In second case, the NER finds one or more NEs in the tweet. These NEs are the candidate SNEs of the tweet. We use three NERs in our experiments. The performance of the NERs on CWC15 dataset is shown in Table 4.4. Though we find that NERs of Ritter et al. and Gimpel et al. give almost equal number of NEs for a tweet, the NEs are not always the same. This is the case with

\[12\] https://github.com/aritter/twitter_nlp
<table>
<thead>
<tr>
<th>Type</th>
<th>Feature</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word</td>
<td>Lower : Change the case of word to lower case</td>
<td>3</td>
</tr>
<tr>
<td>Word</td>
<td>Upper : Change the case of word to upper case</td>
<td>1</td>
</tr>
<tr>
<td>Word</td>
<td>isUpper : Is the word in upper case</td>
<td>2</td>
</tr>
<tr>
<td>Word</td>
<td>isFirstCharHash : True if first character is #</td>
<td>3</td>
</tr>
<tr>
<td>Word</td>
<td>isFirstCharHashOrAt : True if first character is # or</td>
<td>4</td>
</tr>
<tr>
<td>Word</td>
<td>isFirstCharCaps : True if first character is in uppercase</td>
<td>3</td>
</tr>
<tr>
<td>POS</td>
<td>Postag : POS tag returned by Ritter</td>
<td>4</td>
</tr>
<tr>
<td>POS</td>
<td>isStartsWithNN : True if POS tag starts with NN</td>
<td>2</td>
</tr>
<tr>
<td>POS</td>
<td>isStartsWithNNorPR : True if POS tag starts with NN or PR</td>
<td>1</td>
</tr>
<tr>
<td>Chunk</td>
<td>Chunk : Chunk POS tag returned by Ritter</td>
<td>1</td>
</tr>
<tr>
<td>Chunk</td>
<td>isChunkNP : True if chunk POS tag is B-NP or I-NP</td>
<td>3</td>
</tr>
<tr>
<td>Entity</td>
<td>Entity : Entity tag returned by Ritter</td>
<td>4</td>
</tr>
<tr>
<td>Entity</td>
<td>isEntity : True if entity is B-ENTITY or I-ENTITY</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 4.3: CRF Features
Finkel et al. too. So, for our experiments, we combine the three results (discussed in Section 4.3.2 and in Section 4.3.4, design decision D3), to get a super-set of the results of the three NERs.

Here we note that in Section 5.5 where we combine NER outputs, we used two tweetNERs and here we used three. The reason for this is as follows. Ritter et al. and Gimpel et al. are tweetNERs while Finkel et al. (Stan NER) is general for all text. Here we enhanced the NER combination module by adding StanNER also. However not a big jump in NER efficiency was realized with this change as can be seen in Table 4.4.

SNE Identification

The CRF tagger performance in terms of Precision (P), Recall (R) and F-measure (F) in tagging SNE by the BIO encoding is presented in Table 4.5. Tag B-SNE is assigned with a F-measure of 0.63 while I-SNE is assigned with an F-measure of 0.46. Tags B-SNE and I-SNE contribute towards identifying SNE. So we combine the rows of B-SNE and I-SNE as identification of SNE. We get Precision of 0.81, Recall of 0.60 and F-measure of 0.69 in SNE identification. For downstream application B-SNE was used.

Next, we analyze the performance of individual feature types used in the SNE identification. Figure 4.5 shows the graph plotted for the feature types discussed in Table 4.3. Here we see that word features gives best P, R and F in assigning both B-SNE and I-SNE tags.

Here we note that the average span of SNE is same as average span NE from NER as we are using the NER output as the SNE\_candidate. In the CWC dataset we created, the SNE span was about only slightly around 2 words. In other words, most of the SNEs had 2 words, some were single words and few were three.

Evaluation

We evaluate the performance of the SNEIT system in tweet filtering application using a standard dataset for filtering tweets containing relevant NEs. We also compare the performance of SNE identification method with the baseline method for whole tweet linking. Both these evaluations require the NE to be an entity in Knowledge base (KB). So we do entity linking of the tweet. We use Wikipedia as KB to maintain parity with related experiments, i.e the linked entity is a Wikipedia entity. Hence in
this section, we first present linking of SNE in tweet to its Wikipedia entity, followed by evaluating its performance against the baselines.

**Tweet Linking** The SNE identified is linked to a Wikipedia entity\(^{[13]}\). The SNE might represent multiple entities in Wikipedia. In order to disambiguate among the many entities, we use a supervised binary classifier which we call tweet linker. For instance, consider the CWC15 tweet “It seems as if Australians are kicking them out.#AUSvPAK http://t.co/Bpg0tOXwOZ” containing the SNE ‘Australian’ which gets linked to ‘Australia_national_cricket_team’. Notice that though the tweet did not explicitly mention the cricket team, by means of linking, the SNE is disambiguated better.

The tweet linker is trained using the CWC15 tweets that have a Wikipedia title. The dataset is skewed, with the negative samples outnumbering the positive samples (i.e. samples for which Wikipedia title is relevant). Therefore, we focus our attention only on the relevant samples and report the performance for the same in further experiments in this article.

In the experiments on verifying the performance of tweet linker, we keep the NE selection part constant and vary only the linker part. Thus we use the salient NEs annotated in the dataset as the NEs in the experiments in Table 4.3.3 and Table 4.7. The linker part is varied with different KBs or different ranks of a KB.

**KBs** We experiment with three KBs and choose the one that gives best performance in linking the NEs. The three KBs are

1. **Wikipedia** The English Wikipedia dump dated 10 June, 2015 is indexed using Lucene\(^{[14]}\). Using Lucene’s multi-field query, the SNE mention is searched in the Wikipedia title and tweet is searched in the description of the Wikipedia article. The Lucene score gives the context similarity score for the retrieved Wikipedia entities.

2. **DBpedia** DBpedia\(^{[15]}\) is a multilingual KB constructed by extracting structured information from Wikipedia such as info-box templates, categorization information, geocoordinates, and links to external Web pages. For each entity retrieved from DBpedia, we consider the title and abstract. The abstract of the entity is analyzed for similarity with the tweet. This gives context similarity.

3. **GCD** Google Cross-Wiki Dictionary (GCD)\(^{[16]}\) is a string to entity mapping, created using anchor text from various web pages. The strings are the anchor hyper-texts that refer to the Wikipedia page titles. We query the GCD with the candidate SNE. A ranked list of probable Wikipedia entities are retrieved. The ranking criterion is the Jaccard similarity between the anchor text and the tweet. So if the tweet is highly similar to the anchor text, then context similarity will have a high score.

The tweet linker classifies the linking of Wikipedia entities given by KB as correct or wrong. In Table 4.3.3 we compare the performance of three KBs in terms of P, R and F, using a Random Forest (RF) classifier, considering top 1 KB retrieval rank. Here, we find that DBpedia gave better F1 than other KBs. This is probably due to better retrieval ranking in DBpedia and relative freshness of the data.

---

\(^{[13]}\) we use Wikipedia page title and Wikipedia entity interchangeably

\(^{[14]}\) https://lucene.apache.org/

\(^{[15]}\) http://wiki.dbpedia.org/Ontology
since GCD was created in 2012, when number of articles in Wikipedia was 3.8M, whereas DBpedia accesses the current Wikipedia having 4.8M articles.

<table>
<thead>
<tr>
<th>KB</th>
<th>P</th>
<th>R</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wikipedia</td>
<td>0.60</td>
<td>0.45</td>
<td>0.512</td>
</tr>
<tr>
<td>DBpedia</td>
<td>0.81</td>
<td>0.54</td>
<td>0.64</td>
</tr>
<tr>
<td>GCD</td>
<td>0.64</td>
<td>0.58</td>
<td>0.58</td>
</tr>
</tbody>
</table>

Table 4.6: Choice of KB

The tweet linker chooses the top (t) Wikipedia entities given by KB and classifies the linking as correct or wrong. Experiments are conducted with t at 1, 2, 3 and 4. Performance (in terms of P, R and F) for tweet linking on CWC15 dataset is reported in Table 4.7. A RF classifier is used as the linker. Choosing the top 1 Wikipedia entity gives the best result with F-score of 0.64.

**Linker features** The features of the SNE used by the linker are the following:

- **F1. Lexical position or Word Order of NE:** When there are multiple NEs detected in a tweet, this feature captures the index of the SNE in the list of NEs.
- **F2. Presence in Hashtag:** This binary feature indicates if the SNE occurs within a hash tag, i.e if the SNE is wholly or partly present in a hashtag.
- **F3. Preceded by @:** This binary feature indicates if the SNE is preceded by a @ symbol in the tweet, which makes it a Twitter userid.
- **F4. Topic word:** This feature indicates if the SNE contains the topic word. Here topic refers to topic of the tweet collection (discussed in feature F8) (refer design decision D1 in Section 4.3.4).
- **F5. Retrieval Rank:** The Wikipedia and GCD corpus are indexed using Lucene. The rank of the entity is normalized with result size. For DBpedia, this is the retrieval rank of DBpedia query results.
- **F6. Context Similarity:** Context Similarity is calculated as the Jaccard similarity between the tweet and the context of the Wikipedia entity in the KB, as suggested by Dalvi et al [7]. In case of GCD, context is the anchor text from web page. In Wikipedia and DBpedia, context is the abstract of the entity. So Lucene score gives the context similarity.
- **F7. Link Probability:** Probability that the mention is a hyperlink in Wikipedia. A corpus of all hyperlinks in Wikipedia is created. Link Probability is the ratio of number of times a mention occurs in this corpus to the total number of times it occurs in Wikipedia.
- **F8. Salience Probability:** We define salience probability as the probability that the NE is salient to tweets on the topic (Refer design decision D1 on choice of topic in Section 4.3.4). To calculate this we first collect the tweets on given topic. In the case of CWC15, the topic is ‘cricket’. The list of experts in the topic is obtained from the Cognos API [7]. We collect tweets of these experts to create the topic-specific tweet collection as shown in the Figure 4.6. NERs (same as the one used in candidate SNE generation in Section 4.3.2) identify the NEs in the tweets. For a new tweet in this topic, we argue.
that the NEs prominent to the topic are the salient NEs of the tweet. Let \( t \) be a tweet containing NE \( n \) and topic \( c \). Let salience probability or probability that \( n \) is the SNE of a tweet, be referred as \( P_s(n|t) \). We can write this as Equation 4.1

\[
P_s(n|t) \propto P(n|\text{TopicTweetCollection})
\] (4.1)

Assuming independence of the entities in a given tweet Equation 4.1 can be written as Equation 6.3

\[
P_s(n|t) = P(n|c,t) \cdot P(c|t)
\] (4.2)

where

\[
P(c|t) = \frac{N(c)}{\sum_{c'} N(c')}
\] (4.3)

\[
P(n|c,t) = \frac{\sum_{n \in t} N(c)}{N(c)}
\] (4.4)

and \( N(c) \) is the Number of tweets in topic \( c \).

**Figure 4.6: Page rank vector creation**

F9. Page Rank: Page rank computation is shown in the Figure 4.6. The NEs prominent to the topic are \( N \) (created in F8). These NEs are used to create the Page rank vector. Two NEs are linked if the same expert talks about both of them and weight of the link is the Pointwise Mutual Information (PMI) \([?]\) of the NEs.

\[
PMI(N_1, N_2) = \log \frac{n(N_1, N_2)E}{n(N_1)n(N_2)}
\] (4.5)

where \( n(N_{(c,j)}) \) is the number of experts who talked about the named entity \( N_{(c,j)} \) and \( E \) is the total number of experts in this Cognos topic. The page rank vector of SNE (when present) is used as the feature for the learner. Creating the topic-specific tweet collection and page-rank calculation is done as pre-processing.

Now we compare the tweet linking performance by using different classifiers. As seen from the Table 4.8 Support Vector Machines (SVM) classifier performs well even when the size of positive samples were less (28% of the total samples). Random Forest (RF) classifier gave best precision, as also seen in the baseline work of Meij et al. \([?]\) as well as in Yamada et al. \([?]\). Note that the values in Table 4.8 are for ‘1’ classifications (positive samples). Values for ‘1’, ‘0’ and average is presented in Table 4.10.

RepLab dataset We evaluate the SNEIT system using RepLab Filtering task. The RepLab tasks \([?]\) is about monitoring the reputation of entities (companies, organizations, celebrities, etc.) on Twitter. As part of the task, analysts were asked to identify the potential mentions from the stream of tweets and map them to the corresponding entities. Thus the RepLab dataset contains manual annotations of
tweets with entities, annotated with two possible values: related and unrelated. The filtering task is about determining which tweets are related to the entity and which are not. In this evaluation, rather than filtering all tweets containing the entity-of-interest, we filter tweets where entity-of-interest is the SNE, as related.

For a test tweet, we determine the SNE using SNEIT. The SNE is queried on KB. When the KB result ranked @1 is the related entity we count it as an accurate prediction. Accordingly we report the Accuracy (Acc), Reliability (R) and Sensitivity (S). R and S is precision and recall for filtering task. We also report F-measure (F) in Table 4.9. The last row of the table gives overall performance, which is averaged over all queries, of the SNEIT system. The overall F is 0.55 while the median of F value for RepLab filtering task results of 2013 is 0.27 with the highest being 0.49. We see that the SNEIT system performs better than most RepLab participant systems. Columns ‘Tweet Collection Size’ gives number of tweets containing mention of the entity and column ‘Tweets with SNE’ gives number of tweets where the entity is SNE. These prove that SNE is better candidate for filtering task.

Baseline System Meij et al. [? ] propose ‘whole tweet’ entity linking in their work, where they link the tweet to the topic or the entities describing the theme of the tweet. As this is linking a tweet to its salient entity, we consider this work as our baseline system and evaluate the performance of the SNEIT.
system against this system on the CWC15 dataset. Towards this goal, we have re-implemented the Meij et al. system. The comparison is presented in Table 4.10. We have used the Random Forest (RF) classifier and COMMONNESS concept ranking, which is reported to have produced the best results by Meij et al. The features of Meij et al. system used in our re-implementation includes N-gram features (LEN, SLINKPROB), Concept features (INLINKS, WLEN, CLEN), N-gram + concept features (SPR, NCT, TCN, TEN, COMMONNESS) and Tweet features (TWCT, TCTW, TETW, TAGDEF, URL). The average precision and F-measure of this system are 0.85 and 0.87 whereas that of original Meij et al. system were 0.57 and 0.48 respectively.

Table 4.10: Comparison of SNEIT system performance with baseline

<table>
<thead>
<tr>
<th>Method</th>
<th>Label</th>
<th>P</th>
<th>R</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Meij et al.</td>
<td>0</td>
<td>0.92</td>
<td>0.97</td>
<td>0.94</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0.22</td>
<td>0.09</td>
<td>0.13</td>
</tr>
<tr>
<td>Avg.</td>
<td></td>
<td>0.85</td>
<td>0.89</td>
<td>0.87</td>
</tr>
<tr>
<td>SNEIT</td>
<td>0</td>
<td>0.73</td>
<td>0.98</td>
<td>0.84</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0.62</td>
<td>0.09</td>
<td>0.16</td>
</tr>
<tr>
<td>Avg.</td>
<td></td>
<td>0.70</td>
<td>0.73</td>
<td>0.64</td>
</tr>
</tbody>
</table>

The performance of SNEIT system in tweet linking is 62%, while the baseline is 22% in terms of precision. The corresponding improvement in F-measure is 3%. Interestingly there was no difference in recall achieved between SNEIT and Meij et al. system. These results are statistically significant when tested with t-test (1-tail 95% value is 1.645 and 2-tail 95% value is 1.96).

4.3.4 Observations

Performance In this section we discuss the time to execute and memory footprint of SNEIT system across varying test set sizes. Figure 4.7 shows plot of cumulative time taken to execute across varying test set size. Test set consists of 410 tweets (denoted as 100% test set size in the Figure 4.7) from CWC15 dataset. From the figure we can see that the execution time is rising linearly with load test set size. Memory (RAM) usage of SNEIT assessed from the heap space (we used Java code) usage value averages to 449 MB. The process of SNE Identification occurs almost real-time. The CRF features

Figure 4.7: Execution time
are not computationally intensive. All the processing is on-line except page rank (feature $F9$ of tweet linker) vector creation in evaluation. This is done by an in-memory implementation using NetworkX package. On a 16 GB RAM machine, this took close to 3 minutes to create page rank vector for the cricket tweet collection.

Design decisions

**D1 Extraction from single tweet versus extraction from multiple (related) tweets** In their work, Ardon et al. prove that theme or topic based collection of tweets is better suited for information extraction compared to individual tweets. Hence we have used topic based tweet collection in our features topic word ($F4$), salience probability ($F8$) and page rank ($F9$).

**D2 NER as a Candidate SNE Generator** In SNEIT, we make a textual scope assumption i.e. the salient entity is contained in the tweet. Hence, a system that is capable of identifying NEs in the tweet would serve as a candidate generator for a SNE identification system. We test this assumption in the CWC15 dataset. While creating the CWC15 dataset, annotators mark a tweet as N if no NEs were listed. This includes cases where the picture showed an NE and no NE was identified by NERs. We counted all the tweets marked N and found it to be 2.8% of total tweets. In other words, in 97% of the tweets, at least one of the salient entities is in the candidate entity set identified by the NER. Therefore it is reasonable to use the NER as a SNE candidate generator.

**D3 Merging the three NER results** The NER results (set of NEs) could be merged in two ways. First is by taking a union of all the results. For example, \{“ind vs aus”\} U \{“ind vs aus quarter final”, “ind vs aus”\} = \{“ind vs aus”, “ind vs aus quarter final”\}. The other is to merge the NER results, if a NE is a sub-string of another NE as in \{“ind vs aus”\} U \{“ind vs aus quarter final”\} = \{“ind vs aus quarter final”\}. We choose the first as it ensures that a string having multiple NEs does not eliminate the string with a single NE. This method of mention choice is advocated by Gattani et al. Literature gives multitude of surveys comparing NER methods especially openSource NER packages. They conclude that each NER tool is optimized for a type or domain. Hence a generalized NER should use multiple tool and use a pooled (best of all) result.

**D4 Dataset contains only tweets with image** Zhao et al. did a statistical analysis on multimedia content in social media data. They expect the proportion of image-tweets (tweets containing images) to increase. Chen et al. note that image-tweets constitute over 45% of overall traffic in (Chinese) Weibo. With increasing number of image-tweets, a sample of tweets with only image-tweets has higher chances of being a random sample. Further the presence of 507 sarcastic tweets in CWC15 dataset indicates the eclectic nature of tweet authors rather than limited number of tweet sources like news or sports agencies. The higher number of sources increases chances of a random sample. Out of a sample of 2000 tweets, Pear Analytics classified 40% as containing pointless babble, with another 37.55% as merely conversational. They found that only a small fraction contains topics of general interest. In the CWC15 dataset, we found similar figures with only 33% of the tweets as useful. The similar distribution

---

16https://networkx.github.io
of non-informative and informative content across sample containing all tweets and sample containing
only image tweets, shows our tweet sample is a random sample.

D5 Choice of domain The annotator’s background knowledge and interest in the domain of the
tweet text has a positive effect on the quality of the annotation. As the task was selecting the salient
of the NEs, background knowledge of the NEs was needed. Considering our annotator’s interest we
chose the popular sporting event (cricket world cup), a popular entertainment event (annual film award)
and a much awaited product release (apple watch release). Considering the age group of our annotators,
the popularity of these events ensured annotator’s domain knowledge and hence high quality of the
annotation (discussed in Section 4.3.1).

D6 Choice of Cricket World Cup The Cricket World Cup 2015 (CWC2015) held in Australia and
New Zealand created unprecedented levels of online and social media interest, as can be seen from the
of the event organizing body. There were over 26 million
unique visitors to the official cricket world cup website during the first 30 days of the 45 day event.
These visitors made up over 225 million Page views. The video clips of matches of the tournament was
watched by fans from over 200 countries with over 24 million video plays combined across website and
App. On Social Media there was more interest than ever seen before for a global cricket event as people
in millions interacted with the tournament. On Facebook, 36 million people generated 341 million
interactions, with cricket becoming a regularly trending topic throughout February and March. On
Twitter, the discussion around #cwc15 was huge, with over 8 million tweets sent around the tournament,
with over 800 Million live tweet impressions from the group stages. In comparison, the 2012 London
Olympics generated 150 million tweets\footnote{http://thenextweb.com/media/2012/08/13/spice-girls-go-gold-olympic-twit}. Thus the CWC 2015 was one of the most engaging topics on
social media and twitter, producing significant amounts of data convincing us to create a dataset around
this topic.

D7 Choice of tweets (with images) Chen et al. \cite{?} studied why tweeters use image tweets over
text-only ones and note that the preference of posting image tweets or text tweets is correlated with the
content. For example, advertisement tweets tend to include a product image to make it more informative;
whereas tweets about the everyday routine (tweets whose topics are about work or sleeping) are prone to
be text-only. This gives us more reason to look for entities in tweets containing images than in text-only
tweets.

D8 Memes and sarcastic comments in tweets While annotating tweets manually to create the CWC15
dataset, we came across the issue sarcastic tweets. We mark them ‘S’ and use them as negative examples
to learn SNE. We have not used memes in this chapter or elsewhere in the thesis. Memes are an image,
video or piece of text, typically humorous in nature, that is copied and spread rapidly by Internet users,
often with slight variations. Memes are used to convey opinion or sentiment towards a topic or NE of
the topic. Objective of our study was to identify NE and not opinion towards NE or topic. Its hardly
the case that the NE in the meme is the SNE of the tweet. Thus using memes instead of images was not attempted in our (SNEIT) approach.

4.3.5 Conclusion

We described SNEIT, a method for identifying SNEs in tweets. We demonstrated how a simple observation that the image accompanying the tweet captures the salient named entities helps create a dataset for learning SNEs in tweets. We train a supervised sequence labeler with a small and rich set of features on the CWC15 dataset to identify SNEs in a tweet. Salient entities improved the richness of the semantic network in “Web of Things” paradigm [?]. Our experiments showed that only 43% of named entities in a tweet are salient, thereby making it an indispensable task to filter them from non-salient entities. Identifying salient entity benefits a wide array of applications such as tweet filtering, summarization, content-analysis, user modeling and current trends prediction [? ?]. We have made the entire annotated dataset, including 507 sarcastic tweets and the source code of the JSNEIT system, publicly available.

4.4 Semantic Attribute of a Named Entity

In this section we present a method on identifying the category or semantic attribute of an NE. These experiments were conducted on product entities and product titles specifically. Identifying semantic attributes help in studying evolution of product entities, which help analysts understand the change in particular attribute values for products. However, studying the evolution of a product requires us to be able to link various versions of a product in a temporal order. While it is easy to temporally link recent versions of products in a few domains manually, solving the problem in general is challenging. The ability to temporally order and link various versions of a single product can also improve product search engines.

4.5 Identifying Semantic Attributes of Named Entity: Predecessor Version

In this chapter, we propose a method of finding the previous version (predecessor) of a product entity. Given a repository of product entities, we first parse the product names using a Conditional Random Fields (CRF) model. After identifying entities corresponding to a single product, we solve the problem of finding the previous version of any given particular version of the product. For the second task, we leverage innovative features with a Naïve Bayes classifier.

19Source code and dataset at http://tinyurl.com/pqjn8tc
Motivation: A large number of users search on the web for product entities with a variety of intents: knowing product specifications, comparing products, buying products, selling products, reading reviews, etc. Online data on consumer products is also increasing day by day thanks to a large number of e-commerce and review websites. While such e-commerce websites already host information about millions of products, new products and new versions keep appearing frequently. Many of such products have a long history. Linking various versions of the same product temporally can help us understand evolution trends for particular products. Also, the ability to link all versions of the same product together, and the ability to rank them temporally can improve product search engines.

Predecessor Prediction Problem: Given a set of versions of a product entity, one can construct a product version tree by identifying parent-child relationships. Such a product version tree can be constructed by linking a child node to a parent such that the parent product version was released just before the child, and is the closest to the child node in terms of its specifications. Thus, the central problem in creating such a product version tree is to find the immediate predecessor version of a particular product version. We focus on the predecessor prediction problem in this chapter.

Example Usage: Product predecessors have already been used to improve user experience on product portals like Amazon as shown in Figure 4.8. But it is unknown as to how Amazon achieves this. We use such product pairs obtained from Amazon as the golden set for our experiments.

Figure 4.8: “Newer Model” Feature on Amazon

Challenges: Predicting the predecessor version poses the following challenges. (1) There is no common convention followed in naming versions or products. Even the same manufacturer does not follow any standard convention. Information extraction from short listing titles present a unique challenge, with the lack of informative context and grammatical structure. (2) Product descriptions are mostly provided in unstructured natural language form [? ]. Product mentions in the description do not follow any canonical
name. For example, the product *JVC S-VHS Camcorder* may appear as *Super VHS Camcorder* or *S VHS Camcorder* in the product description. Linking such variations is challenging.

### 4.6 Approach to Predicting Predecessor Version Problem

We solve the problem of finding the predecessor version of a given product version in two stages. In the first stage, given a set of product entity names, we parse the product names to identify the “brand”, “product”, and “version” indicating words from the product name. The first stage thus gives us all the product entities belonging to the same product as the given product version. The second stage ranks these candidates and chooses the most probable predecessor version from the candidate set. We discuss the supervised approach for both the stages in this section.

### 4.7 Experiments on Predicting Predecessor Version Problem

**Parsing the Product Title**

Product titles and descriptions used for our experiments was crawled from Amazon. The data for every product has the title and other details like the product ID, product description, reviews, etc. Typically a product title consists of product brand, product name, product attributes, attribute values, version information, and accessory words. We aim at labeling such product titles with the following tags: brand name, product name, version and others. For example, a product in this dataset has product title *Leica D-Lux 6 digital camera*. In this example, the *Leica* will be labeled as “brand name”, *D-Lux* will be labeled as “product name”, *6* will be labeled as “version” and *digital camera* will be labeled as “Others”.

The task of parsing the product title translates to categorizing words in the product title as brand name, product name, version and others. Typically a product title consists of “brand” before “product” with a high probability. Thus, the order of words in a product title follows a sequential pattern. Hence, we choose a CRF based approach to solve this problem. CRFs are a class of statistical modeling method and are widely used for labeling or parsing of sequential data. The CRF is trained using manually labeled data for a few product titles. The CRF tagger can be trained on the (word, label) pairs using features obtained from the product description, context patterns surrounding the labels and linguistic patterns frequently associated with the labels in the training set. It is then used to label words in product titles from the test set. We used the MALLET [? ] toolkit.

Next, we will discuss the set of features we use for the CRF model. We use three main types of features. Consider a product title *Apple iPad Mini (White)*. We will use this example to describe a few of the features.

**Linguistic features**

We analyze the Part-Of-Speech (POS) tags of the words in the product title to identify POS tag patterns. We check if the words of the product title have POS tag in the set \{NNP, MD, VB, JJ, NN, 51
We have a binary feature for presence or absence of each of these parts of speech. For example, the word *White* in the above example will have the linguistic feature JJ set as TRUE and all other linguistic features set as FALSE.

**Context and Word Characteristics Features**

The contextual text of the brand name, product name and version in the product title conveys valuable information. For each word token, we define the following features: (1) Position of the word from the beginning of the product title, in terms of number of words; (2) Is the word the last word in the title; (3) Is the word alphabetic; (4) Does the word represent a color; (5) Is the word numeric; (6) Is the word surrounded by parenthesis; (7) Is the previous word “for”.

In the above example, the word *White* has the following context features set to TRUE: the word is alphabetic, the word is surrounded by parenthesis, the word represents a color, and the word appears at position 4 in the title. If one has an exhaustive dictionary of all brand names, features like “Is the word a brand?” could also be considered.

**Product Description Features**: Product description specifies the attributes and the values for these attributes of the entity. We have a binary feature corresponding to every attribute, which checks for presence or absence of that attribute’s value in the product description. We identify nine such features as follows: description itself, weight, review date, review title, model, category, bought next, bought along with, URL.

We propose to label the product titles using a CRF model based on the above set of features. Next, we group together product entities that have the same brand name and product name but differ only in the version part. Given any query product version, we identify its (brand, product) cluster after labeling the query product entity title using the CRF. All members of this cluster are candidates for being its predecessor version.

**Predicting Predecessor Version**

As discussed in the previous subsection, the first stage provides a set of product entities with the same brand name and product name as compared to the query product entity. These are candidates for the second stage. The second stage as discussed in this subsection uses a classification based approach to identify the most probable product predecessor version from the candidate set for the query product entity.

The classification approach relies on the following intuitive set of binary features.

**Lexical Ordering**

This feature indicates whether the candidate lexically precedes the given query product version. The version names often manifest alphanumeric patterns. For example, “Nokia Lumia 1020” precedes “Nokia Lumia 1320” both with respect to time of release as well as lexicographically. Hence, product versions can be ordered lexicographically. We would like to note that lexicographic ordering is beyond alphabetical ordering, i.e., in addition to alphabetical order, we consider numerical order also. So we call lexicographical order. Due to the numerical ordering ‘Nokia Lumia 1020’ precede ‘Nokia Lumia 1320’.
Review-Date Based Ordering

This feature indicates whether the candidate is older than the given query product version. Products can be ordered temporally based on their earliest review date. We obtain posting date of the earliest review for all products in our dataset. We then determine the age of a product version using the review date of its earliest review. Intuitively, the version which got reviewed earlier has a higher probability of being older.

Mentions Based Ordering

This feature indicates that the candidate product version was mentioned in the given query product version’s description or reviews. When describing a new version of any product, the product manufacturers often highlight improved features in this version compared to the previous version. Similarly, reviewers often compare the current version with the previous version when writing reviews. We use these mentions of product names in product reviews and description towards linking two versions of the same product. We use simple substring matches (with the product name) to link the mentions with the product entities.

Given a query product version, a classifier is used to rank all the candidate product versions and the most probable one is chosen.

4.7.1 Dataset

We crawled ~462K product description pages from Amazon. These pages were parsed to obtain product title, product description, reviews, etc. The dataset is publicly available

4.7.2 Experiments and Results

Parsing the Product Title

From the dataset, we selected 500 product titles from the camera & photo category, and hand labeled the words as brand name, product name, version or others. A CRF sequential learner was trained on this dataset to predict the labels. The Precision (P), Recall (R) and F-measure (F) from a five fold cross validation of this task is presented in Table 4.11. Note that the label brand name was identified with the highest precision. Even the product name was identified with a good precision. This implies that given a query product version, we obtain its candidate product predecessor versions with a high accuracy in the first stage. However, the precision for the version label is quite low.

An interesting observation we made was that the classifier incorrectly marked some product name words as version and vice versa. This was because for such product entities, the product name and the version indicator does not appear as two different words; instead it appears as a single word. For example, ge dv1-co and foscam fi8918w. In such cases, the words dv1-co and fi8918w serve both as the version indicators as well as product names. Hence, we also trained a CRF model for a

20http://tinyurl.com/lclapy8

53
Table 4.11: CRF Accuracy for the Product Title Parsing Task (when `product name` and `version` are treated as Separate Labels)

<table>
<thead>
<tr>
<th>Label</th>
<th>P</th>
<th>R</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>brand name</td>
<td>0.98</td>
<td>0.65</td>
<td>0.77</td>
</tr>
<tr>
<td>product name</td>
<td>0.89</td>
<td>0.58</td>
<td>0.69</td>
</tr>
<tr>
<td>version</td>
<td>0.69</td>
<td>0.48</td>
<td>0.55</td>
</tr>
<tr>
<td>others</td>
<td>0.84</td>
<td>0.98</td>
<td>0.91</td>
</tr>
</tbody>
</table>

Table 4.12: CRF Accuracy for the Product Title Parsing Task (when `product name` and `version` are treated as the Same Label)

<table>
<thead>
<tr>
<th>Label</th>
<th>P</th>
<th>R</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>brand name</td>
<td>0.98</td>
<td>0.65</td>
<td>0.77</td>
</tr>
<tr>
<td>product name/version</td>
<td>0.88</td>
<td>0.55</td>
<td>0.67</td>
</tr>
<tr>
<td>others</td>
<td>0.85</td>
<td>0.98</td>
<td>0.91</td>
</tr>
</tbody>
</table>

three-class labeling task with labels as `brand name`, `product name/version` and `others`. The precision, recall and F1 score for this three-class labeling task are presented in Table 4.12.

Thus if the 4-label CRF identifies product name and version as separate words, we use the output from the 4-label CRF, else we use the output from the 3-label CRF. For the output from 4-label CRF, clustering will be performed on (brand name, product name) to cluster and obtain candidates for the query product version. For the output from the 3-label CRF, clustering is performed on the brand name alone.

Predicting Predecessor Version

As shown in Figure 4.8, Amazon lists the immediate preceding version of a product, for some products. For example the product description page of Canon Powershot A550 7.1MP Digital Camera with 4x Optical Zoom lists Canon Powershot A470 7.1MP Digital Camera with 3.4x Optical Zoom as the immediate preceding model. We collected all such pages from our dataset and extracted (version1, version2) ordered product pairs. We use this as our golden truth for the product immediate predecessor prediction task. 40 out of our 500 product pages have such golden predecessor version mentions.

Using the output from the CRFs, we propose to cluster the product entities based on the brand name. For each cluster (i.e., brand), product versions in the golden truth will be considered. For each product version, list every possible candidate of the predecessor product version. Such (product query, candidate) pairs will be used to define instances for the Naïve Bayes classifier which is then used to compute the probability that the candidate could be a predecessor of the queried version. Note that for this task, the dataset is highly imbalanced. The positive class (the correct (query, predecessor) pairs) is in minority.

Table 4.13 presents the results of the product predecessor prediction for the positive class in terms of the following metrics: True Positive Rate (TP), False Positive Rate (FP), Precision (P), Recall (R) and

\[ \text{TP} \]
\[ \text{FP} \]
\[ \text{P} \]
\[ \text{R} \]

\[ ^{21}\text{These titles are listed in the file “goldenData” in the dataset at http://tinyurl.com/lclapy8} \]
Table 4.13: Classifier Accuracy for the Positive Class for Product Predecessor Version Prediction

<table>
<thead>
<tr>
<th>Feature</th>
<th>TP</th>
<th>FP</th>
<th>P</th>
<th>R</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lexical + Review-Date</td>
<td>0.632</td>
<td>0.049</td>
<td>0.533</td>
<td>0.632</td>
<td>0.578</td>
</tr>
<tr>
<td>All features</td>
<td>0.579</td>
<td>0.049</td>
<td>0.512</td>
<td>0.579</td>
<td>0.543</td>
</tr>
<tr>
<td>Review-Date</td>
<td>0.579</td>
<td>0.061</td>
<td>0.458</td>
<td>0.579</td>
<td>0.512</td>
</tr>
<tr>
<td>Review-Date + Mentions</td>
<td>0.553</td>
<td>0.047</td>
<td>0.512</td>
<td>0.553</td>
<td>0.532</td>
</tr>
<tr>
<td>Lexical + Mentions</td>
<td>0.5</td>
<td>0.049</td>
<td>0.475</td>
<td>0.5</td>
<td>0.487</td>
</tr>
<tr>
<td>Lexical</td>
<td>0.5</td>
<td>0.056</td>
<td>0.442</td>
<td>0.5</td>
<td>0.469</td>
</tr>
<tr>
<td>Mentions</td>
<td>0.45</td>
<td>0.049</td>
<td>0.462</td>
<td>0.45</td>
<td>0.456</td>
</tr>
</tbody>
</table>

F1 score (F). The results were found to be statistically significant (95% confidence). Each row indicates the feature sets used. Note that none of the orderings individually could provide reasonable accuracy. Each of the orderings have drawbacks when used individually. For example, lexical ordering does not work in cases where no such ordering is followed. For example, Google Nexus 7 was the earlier version of Google Nexus 5. Review-Date based ordering does not work for those products which was launched on Amazon later than their actual release dates. Similarly, Mentions based ordering does not work if the review text or product description does not contain mentions of previous versions, or the mentions could not be detected accurately. This stresses the need of combining various approaches for this task. The combination of the Review-Date Based Ordering and the Lexical Ordering performed the best in predicting the correct product predecessor version entity (~53% precision). Note that this precision value is good, because we have reported numbers on the minority class only.

4.7.3 Observations

Besides the review-date based ordering, other interesting ways of ordering product versions as follows.

- The release date of product versions could be extracted from their manuals. But manuals are not easily available for most products, while reviews are in plenty.

- We queried a search engine with the brand name and the product name. Based on the order of the search results, assuming that the later versions appear at the top, one can rank product entities. But both the coverage as well as the precision of such an approach was found to be quite low.

4.7.4 Conclusion

In conclusion, we proposed the novel problem of predicting the predecessor version of a given query product version. We presented the proposed two-stage approach to solve the problem: (1) parsing the product title, and (2) predicting the predecessor given all candidates. Experiments on a dataset crawled from Amazon show that our methods achieve a precision of ~88% for the first stage, and a precision of ~53% for the second stage. The solution can be helpful for a variety of applications like building product...
version trees, studying entity evolution, product search engine ranking, comparing product versions, etc. Though we tested the method for the camera & photo category only, the method is generic. Product attribute values usually show a trend across product evolution. For example, pixel resolution of “Canon Powershot cameras” has been increasing, weight of “Samsung hard disks” has been increasing over time.

4.8 Conclusion

In this thesis we are looking at ways to improve Entity Linking and Knowledgebase Expansion in a joint manner. We start with ‘mention detection’ sub task of EL which we elaborated in this chapter. The text under consideration for EL typically talks about many NEs. Linking all the NEs mentioned in text may not be helpful [?]. Named Entity Extractor selects NE mentions to link based on importance or salience of the NE to the document and if the NE is semantically related to other NEs mentioned in the document. In this chapter we presented two methods to assess the importance of NE in the text. First method predicted ‘salient’ NEs of the document from the NE’s syntactic and semantic structure. Second method predicted importance of NE by its relation to other NEs in the text. Having chosen the NE mentions to link, we look ways to link the NE mention to KB entity in the next chapter.
Chapter 5

Linking the Named Entity mentions to Knowledge Base

In this chapter, we explain how Named Entities (NE) are extracted for Knowledge Base enhancement by Entity Linking. Extracting NEs for enhancing Knowledge Base (KB) consists of two steps (i) identifying the NEs in the text and (ii) identifying the KB entry of the NE. The KB stores facts about the NE like name, type, attributes and relationship with other entities in the KB. Name is the canonicalized name of the entity. For example, the phrases “Trump” and “President Trump” refer to the canonicalized name “Donald_Trump”. They are linked to the entity “Barack_Obama” in the KB.

Identifying the phrases “Obama” and “President Obama” as NE is Named Entity Recognition (NER). Basically it is identifying if the NE can potentially represent an entity in the KB. However a NE can represent many entities in KB. For example, the NE “Stanford” may refer to “Stanford University” or the place “Stanford”. Figuring out the NE refers to which entity among the entity options in the KB is called disambiguation. Disambiguation of NE is done by means of linking it to the corresponding KB entity.

5.1 Entity Linking performance on varying text structure and context length

The textual context of the NE plays an important role in the disambiguation of the NE. We have seen the dependence of EL performance on the context availability in ‘C2 Context as aid’ in Section 1.4. However with text size reducing from documents (web-pages and blogs) to short-text (tweets and queries) in the ever emerging web, the size of textual context has also reduced. This was further accentuated by decreasing structure of the text, with EL performing well on documents with good semantic and syntactic structure like web logs or news articles to EL performing poorly in documents like web queries which lack strict syntactic and semantic structure. This was illustrated in Figure 1.3. This has led to increased emphasis on techniques to improve inference from the (lesser) available context. Accordingly we have different approaches for linking the NE, for different context lengths and content structure.
5.2 Entity Linking on Documents

The text of a document has a well defined grammatical structure in contrast to social media generated text. So we call the text of document as ‘clean’ or ‘structured’. The structure aids in prediction of NEs in the text, with very high confidence.

5.3 Approach to Entity Linking on Documents

The task of EL is to determine for each query entity, the KB node being referred or if the query entity is not present in the KB. The query consists of a named entity and an associated document-id. The document-id refers to the Document Collection (DC) using which we need to link the named entity to its corresponding node in the KB, if any. The purpose of the associated document is to provide context that might be useful in linking it. We need to return the entity node if the query entity is present in the KB else NIL.

Entity linking is challenging for three primary reasons. First is polysemy, where an entity name is shared by different entities. This is also referred as ambiguity and involves disambiguating the given name among the possible meanings. Second is synonymy where entities are referred to by different name variants or aliases. This is also referred as variability and involves reconciling all instances or mentions of an entity in text and all of its variants. The third problem is identifying when an entity mentioned in text is does not have a node (entity) in the KB. The entity mention in such cases is called a NIL entity. Detecting NIL mentions is important not only to avoid creating spurious links, but also for identifying new candidates for addition to the KB.

In this section, we show how the above three challenges are solved by the SIEL TAC 2014 EDL system. The SIEL TAC 2014 EDL System, is the Entity Discovery and Linking (EDL) system was built for the TAC KBP 2014 EDL challenge. The challenge involves extracting named entities from a source collection of English textual documents, and linking them to TAC KB. It also identifies NIL entities.

An overview of the SIEL TAC 2014 system is presented in Figure 5.1. It consists of four sequential (serial) elements.

A. Named Entity Recogniser (NER): The first element is a Named Entity Recogniser (NER). We use a Stanford NER module which uses surface and contextual evidence of text to labels sequence of words in a text which are the names of things (such as person and company name) and predict the entity types for the mention. The entity types identified are PERSON, LOCATION and ORGANIZATION.

B. Named Entity Disambiguator (NED): The entity mention identified by NER could be polysemous. We resolve polysemy with a Named Entity Disambiguator (NED). For a mention \( m \), the NED computes all the N possible entities \( e \) that \( m \) could be disambiguated to.

\[
e = [e]_{n=1}^{N}
\]  

(5.1)
NED uses two database derived from the TAC-KB for this. The TAC KB was pre-processed to create two indices as below

*Link DB*: The TAC KB contains a set of entities, each with a canonical name and title for the Wikipedia page, an entity type, an automatically parsed version of the data from the infobox in the entity’s Wikipedia article, and a stripped version of the text of the Wiki article. The Wikipedia infoboxes and entries are taken from an October 2008 snapshot of Wikipedia. Wikipedia article text frequently contain links to other Wikipedia pages. As an extension the text content of the TAC KB nodes also frequently contain links to other TAC KB nodes. The anchor text of the link to Wikipedia page is a good human-edited representation (or explanation) of the Wikipedia page it points to. Many works in Entity Linking and linking to Wikipedia exploit this link structure of Wikipedia to do disambiguation. In these lines we extracted all links between KB nodes to create two indices.

- **Anchor Index**: This is a map of Anchor tag to KB-ids. It was implemented with a MongoDB. It is used to calculate the Link Probability which is the probability that the given word (phrase) occurred as a link in the KB.

- **In-Link Index**: This is a map of KB-id to List of KB-ids, where KB-id is the in TAC-KB entityIds. It uses MongoDB and is used in calculation of bias or Prior Probability which is probability that the given KB-id will link to a KB-id based on its previous occurrences.

*TAC KB*: The whole of TAC KB (including wiki-text) was indexed using Lucene. On querying this index for a word or phrase, Lucene index provides all KB nodes that are related to the word and a score which showed the relevance the retrieved KB node for that word. We used the Lucene score as a feature to train the EDL training model. Further Lucene also indicates if the word (phrase) was found in the title/infobox/text/category of the KB entry. Accordingly position rank is calculated, which is also used as a feature in EDL training model.

Using the indices the NED produces disambiguations(d(.)) for each mention (m), as a set N possible entities e. For each e the feature values(F) namely Link probability, Prior probability, Lucene score and
Position rank, corresponding to $e$ is also computed.

$$d(m) = [e, F]_{n=1}^N$$  \hfill (5.2)

C. **Classifier:** The logistic regression classifier is used to combine to four feature values as the confidence to predicting the entity for the given mention. We used a supervised binary classifier, indicating 1 for correct classification and 0 for incorrect classification (mention and entity do not match). For training, we used named entities identified from the TAC training document collection (LDC2014E15). We extracted link DB and TAC KB based features. An SVM classifier is trained using these four features.\footnote{We used a libSVM implementation in Java.} The classifier combines the feature values as the prediction confidence value.

D. **Linker:** The confidence value is the confidence of classifier for predicting the KB entity for a given mention. All the predicted entities are ranked according to the confidence value and the entity predicted with highest confidence is chosen. Since the data is highly biased towards the 0(negative) class, we obtained very low confidence scores for 1(positive) cases. The confidence of prediction is below a threshold $\tau$\footnote{We chose $\tau = 0.25$ after observation} or if no entity was predicted (all of $e$ is classified as 0), the linker links the mention to NIL.

### 5.3.1 Dataset

In our experiments on EL on documents, we use the dataset created and distributed in Text Analysis Conference \texttt{http://www.nist.gov/tac/} workshops. The Text Analysis Conference (TAC) is a series of workshops that provides the infrastructure for large-scale evaluation of Natural Language Processing technology. TAC comprises sets of tasks known as "tracks," each of which focuses on a particular sub-problem of NLP. We are interested in the Knowledge Base Population (KBP) track, which is aimed to develop and evaluate technologies for populating knowledge bases (KBs) from unstructured text.

The TAC participants are provided with a document collection, a KB (TAC KB) and training and test EL queries. The document of the collection are drawn from news-wire, discussion forums and blog collections. We extract and store the body text, discarding markup and non-visible content if they are formatted using a markup language. The TAC KB contains a set of entities, each with a canonical name and a title for the Wikipedia page, an entity type, an automatically parsed version of the data from the Infobox in the entity’s Wikipedia article, and a stripped version of the text of the Wikipedia article. The Wikipedia Infoboxes and entries are taken from an October 2008 snapshot of Wikipedia. A total of 818741 entities exist in this KB, where each entity is assigned to one of four types as listed in Table 5.1.

EL queries typically consists of entity linking annotations. The NEs are annotated with gold standard entities in the KB. The gold standard is a reference to a TAC KB node or NIL, if there is no corresponding node in the KB. The TAC 2009 train set had 3904 EL annotations in which 43% had valid KB node and
Table 5.1: TAC KB Entity Distribution

<table>
<thead>
<tr>
<th>Entity Type</th>
<th>Type Name</th>
<th># of Entities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Person</td>
<td>PER</td>
<td>114523</td>
</tr>
<tr>
<td>Organization</td>
<td>ORG</td>
<td>55813</td>
</tr>
<tr>
<td>Geopolitical</td>
<td>GPE</td>
<td>116498</td>
</tr>
<tr>
<td>Unknown</td>
<td>UKN</td>
<td>531907</td>
</tr>
</tbody>
</table>

57% had NIL. In the following years, the absolute number of annotation has varied but the relative distribution of KB node and NIL node has remained similar.

5.3.2 Results and Evaluation

We submitted one run to EDL track. However due to an error in calculation of mention offset (we did not count newline characters but the official evaluator counts it), our the mentions we detected did not match with that of official results. Hence we are unable to present the results of our run here. However in the development test dataset (LDC2014E54) our system obtained a Precision (P) of 0.788 and Recall (R) of 0.19, while that of best participant was P of 0.717 and R of 0.642, by the strong typed link match metric.

5.3.3 Observations

Our system resembles the problem of disambiguating the Named Entity(NE) where only limited amount of entity information is present. We built an EDL system from the limited amount of data possible. We tried not to use any external resources like Freebase or DBPedia. We did not access the Web during the evaluation. We did not use offsets in the query. We have also not used any separate Wikipedia dump other than the TAC KB. We used the wiki-text element of reference KB. We have used the following publicly available software packages: Stanford POS Tagger, Stanford NER, Lucene Indexer, LibSVM, MongoDB.

5.4 Entity Linking in Tweets

Social media networks like Twitter have emerged to be major platforms for sharing information in form of short messages (tweets). Analysis of tweets can be useful for various applications like e-commerce, entertainment, recommendations, etc. Entity linking is the one such analysis task which deals with finding correct referent entities in the knowledge base for various mentions in the tweet. Entity linking in social media is important as it helps in detecting, understanding and tracking information about an entity shared across social media.
Entity linking consists of two different tasks, NE detection and NE disambiguation. Entity linking from general text is a well explored problem. Existing entity linking tools are intended for use over news corpora and similar document-based corpora with relatively long length. But as tweets lack sufficient context, these context-based approaches fail to perform well on tweets.

In this section, we describe our proposed approach for EL on tweets. The dataset was distributed as part of the NEEL Challenge 2014 [? ]. The proposed approach disambiguates the entity mentions in the tweets based on three different measures: (1) Wikipedia’s context based measure ([5.5]); (2) anchor text based measure ([5.5]); and (3) Twitter popularity based measure ([5.5]).

NE detection is done using two Twitter part-of-speech (POS) taggers [? ? ].

5.5 Approach to Entity Linking in Tweets

![Figure 5.2: EL on tweets - Approach](image)

The NEEL Challenge task required participants to build semi-automated systems in two stages:

- Named Entity Extraction (NEE) - to extract NE mentions from a tweet; and
- Named Entity Linking (NEL) - each entity extracted is linked to an English DBpedia v3.9 resource

NE Detection NE detection is the task of finding NEs in the given text. We assume that NEs are present in the tweets. Various approaches for named entity recognition in tweets have been proposed recently [? ? ]. This includes spotting continuous sequence of proper nouns as named entities in the tweet. But sometimes named entities like ‘Statue of Liberty’, ‘Game of Thrones’ etc. also includes tokens other than nouns. To detect such NEs, Ritter et al. [? ] proposed a machine learning based system for named entity detection in tweets. Gimpel et al. [? ] present yet another approach for POS tagging of tweets.
We tried both of these POS taggers to extract proper noun sequences. In our experiments Ritter et al.’s tagger gave an accuracy of 77% while Gimpel et al.’s tagger gave an accuracy of 92%. So we merged the results from both as shown in Fig 5.2. The tweet text is fed to the system and the longest continuous sequences of proper noun tokens detected using the above approach are extracted as the NEs from the given tweet. The merged system provided an accuracy of 98% in predicting NEs.

NE Disambiguation Entity disambiguation is the task of assigning the correct referent entity from the knowledge base to the given NE. We disambiguate the NE using three measures as described below. The scores from these three measures are combined using LambdaMART model to arrive at the final disambiguated entity.

Wikipedia’s Context based Measure (M1) This measure disambiguates NE by calculating the frequency of occurrence of the NE in the Wikipedia corpus. Wikipedia’s context based measure has been used in various approaches for disambiguating NEs in tweets. We query MediaWiki API with the NE. MediaWiki API returns the candidate entities in the ranked order. Each candidate entity is assigned its reciprocal rank as score. Thus, a ranked list of candidate entities with their scores are created using M1.

Anchor Text based Measure (M2) Google Cross-Wiki Dictionary (GCD) is a string to concept mapping, created using anchor text from various web pages. A concept is an individual Wikipedia article, identified by its URL. The text strings constitute the anchor hyper-texts that refer to these concepts. Thus, anchor text strings represent a concept. We query the GCD with a NE along with the tweet text. Based on the similarity to the query string, a ranked list of probable candidate entities are created (which is the ranked list using M2). The ranking criteria is based on Jaccard similarity between the anchor text and the query. So if the NE is highly similar to the anchor text, then the corresponding concept will have a high score.

Twitter Popularity based Measure (M3) Tweets about entities follow a bursty pattern. Bursty patterns are the bursts of tweets that appear after an event relating to an entity happens. We exploited this fact and tried to measure the number of times the given NE refers to a particular entity on Twitter recently. The NE is queried on Twitter API and the resultant tweets are analyzed. All the tweets along with the NE are then queried on the GCD and the candidate entities are taken. Based on the scores returned using GCD, all the candidate entities are ranked (which is the ranked list using M3). As Twitter popularity based measure captures the people’s interests at a particular time, it works well for entity disambiguation on recent tweets. In essence, the methods M2 and M3 are similar but with different inputs. Both use GCD, and produce candidate entities and score as output. However, M2 takes NE and single tweet text as input whereas M3 takes NE and multiple tweets as input.

We have three rankings available using M1, M2, M3. Now the task is to arrive at the final ranking of the candidate entities by combining the rankings of the three different models. The rankings of different models should be combined such that the overall F1 score is maximized. For this, we use LambdaMART

---

3[https://www.mediawiki.org/wiki/API:Search](https://www.mediawiki.org/wiki/API:Search)

4[https://dev.twitter.com/docs/api/1.1/get/search/tweets](https://dev.twitter.com/docs/api/1.1/get/search/tweets)
which combines LambdaRank and MART models. LambdaMART creates boosted regression trees for combining the rankings of the three different systems.

5.5.1 Dataset

The dataset was distributed to participants of “Named Entity Extraction & Linking (NEEL) Challenge in #Microposts2014 workshop at WWW14. It can be downloaded. Each data point consists of a tweet, the NE token and the dbpedia URL of NE. The training set has 2,340 tweets with 41,037 tokens and 3,819 named entities. Test set has 1,165 tweets with 20,224 tokens and 1,458 named entities.

5.5.2 Results

The dataset comprises of 2.3K tweets each annotated with the entity mention and its corresponding DBpedia URL. We divided the dataset into the 7:3 (train:test) ratio. Table 5.2 shows the results obtained using the NEEL Challenge evaluation framework. The best results are obtained when a combination of all the measures were used for disambiguation. A 5-fold cross validation on the dataset gave an average F1 of 0.52 for M1+M2+M3.

<table>
<thead>
<tr>
<th>Measure</th>
<th>F1-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>M1</td>
<td>0.355</td>
</tr>
<tr>
<td>M2</td>
<td>0.100</td>
</tr>
<tr>
<td>M3</td>
<td>0.194</td>
</tr>
<tr>
<td>M1+M2</td>
<td>0.355</td>
</tr>
<tr>
<td>M2+M3</td>
<td>0.244</td>
</tr>
<tr>
<td>M1+M3</td>
<td>0.405</td>
</tr>
<tr>
<td>M1+M2+M</td>
<td>0.512</td>
</tr>
</tbody>
</table>

Table 5.2: Results: M1 represents Wikipedia’s Context based Measure (§2.2.1), M2 represents Anchor Text based Measure (§2.2.2) and M3 represents Twitter Popularity based Measure (§2.2.3)

5.5.3 Observations

For effective entity linking, mention detection in tweets is important. We improve the accuracy of detecting mentions by combining various Twitter POS taggers. We resolve multiple mentions, abbreviations and spell variations of a named entity using the Google Cross-Wiki Dictionary. We also use popularity of an entity on Twitter for improving the disambiguation. Our system performed well with a F1 score of 0.512 on the given dataset.

http://ceur-ws.org/Vol-1141/
submitted as Agglutweet_1.tsv
5.6 Entity Linking in Queries

Annotating queries with entities is one of the core problem areas in query understanding [? ]. While seeming similar, the task of entity linking in queries is different from entity linking in documents and requires a methodological departure due to the inherent ambiguity of queries. In this section, we describe our proposed approach for EL on queries. The approach was implemented in the system submitted by our team for the Entity Recognition and Disambiguation (ERD) challenge [? ] organized as part of SIGIR 2014 [http://web-ngram.research.microsoft.com/ERD2014/] to promote research in developing large scale entity recognition and disambiguation algorithms [? ].

5.7 Approach to Entity Linking in queries

In this section we describe the details of entity linking system for queries. The proposed work derives motivation from the work proposed by Ferragina et al. [? ] for linking entities over short documents. Wikipedia inlinks are explored for detecting and linking the entities present in search queries. The NEs are detected based on the probability that the NE appears as an anchor link on Wikipedia. The disambiguation is done based on the probability of the NE linking to a particular entity in KB and on Wikipedia-based semantic relatedness measure as proposed by Milne et al. [? ]. Pruning is done based on the coherence among the entities detected within the text document. The proposed approach which is optimized for both effectiveness and efficiency is explained in detail in the next section.

The system consists of three main components: NE detection, NE disambiguation and anchor pruning. Before discussing these, we discuss a few Wikipedia-based measures which are an essential part of the system components, and also detail few pre-processing steps.

Wikipedia-based Measures The following measures are calculated using Wikipedia’s hyperlink structure. They are used in detecting, disambiguating and refining the annotations of a text with Wikipedia articles.

- **Wikipedia Link-based Measure** ($\delta$): Wikipedia’s extensive network of cross-references provide a huge amount of explicitly defined semantics. Anchors are terms or phrases in Wikipedia articles, to which links are attached. Each link is a manually-defined connection between anchor and its disambiguated concept. Wikipedia provides millions of these connections. Anchors are used to identify candidate concepts for NEs. Wikipedia’s documentation dictates that any term or phrase that relates to a significant topic should be linked to the article that discusses it. Utilizing this article-link-anchor collection, Milne et al. [? ] proposed Wikipedia Link-based Measure for obtaining similarity between two pages $p_a$ and $p_b$ as shown in Eq. 5.3

$$\delta = \frac{\log(\max(|in(p_a)|, |in(p_b)|)) - \log(|in(p_a) \cap in(p_b)|)}{\log(W) - \log(\min(|in(p_a)|, |in(p_b)|))}$$ (5.3)
where \( in(p_a) \) and \( in(p_b) \) are the set of Wikipedia pages pointing to the pages \( p_a \) and \( p_b \) respectively. \( W \) is the total number of pages in Wikipedia.

- **Senses of Anchor \((Pg(a))\):** Wikipedia provides a vast number of anchor texts which capture the various senses they represent. For example, ‘plane’ links to different articles depending on the context in which it is found, and ‘plane’, ‘airplane’ and ‘aeroplane’ are all used to link to the same article. Because of this polysemy, the same anchor, \( a \) may occur in Wikipedia many times pointing to many different pages. We denote this set by \( Pg(a) \).

- **Frequency of Occurrence \((freq(a))\):** \( freq(a) \) denotes the number of times a phrase \( a \) occurs in Wikipedia (as an anchor or not).

- **Frequency of Occurrence as Link \((link(a))\):** \( link(a) \) indicates the number of times the phrase \( a \) occurs as an anchor in Wikipedia. Clearly, \( link(a) \) is always less than or equal to \( freq(a) \).

- **Prior Probability \((Pr(p|a))\):** The prior probability score is the ratio of number of times the anchor links to a particular page \( p \) to the total number of times the NE is used as an anchor in Wikipedia. It is the probability that occurrence of \( a \) is an anchor pointing to the Wikipedia page \( p \).

- **Link Probability, \( lp(a)\):** \( lp(a) \) denotes the probability that an occurrence of a phrase \( a \) is used as an anchor in Wikipedia. It is the ratio of \( link(a) \) to \( freq(a) \).

### Data Preprocessing

The English Wikipedia dump is processed to create three indexes to facilitate the calculation of the measures described in the previous subsection.

- **Anchor Dictionary:** Anchors occurring in Wikipedia are indexed to efficiently compute the link frequency \( link(a) \), total frequency \( (freq(a)) \), pages linking to \( a \) \((Pg(a))\) along with their prior probability \((Pr(p|a))\). The pages are sorted in the descending order of their prior probabilities for faster computations.

- **WikiTitlePageId Index:** This index is used for maintaining the mapping between Wikipedia titles and the corresponding PageIds.

- **In-Link Graph Index:** Each Wikipedia page title is indexed to indicate the list of Wikipedia titles linking to this page.

### NE Detection

NE is word or group of words that could potentially identify an entity in the knowledge base. In this module we identify NEs using the link probability, \( lp(a) \). For detecting the NEs, we take continuous word sequences of up to 6 words and find if the string appears as an anchor in Wikipedia. If the probability of being an anchor is greater than a predefined threshold, then the anchor is taken as a detected NE and all the pages referred by it are taken as possible candidates for the detected NE.

However this involves a large number of look-ups on the Wikipedia anchor index. In order to reduce the number of look-ups, we used two NE filtering methods as follows.
- **Stop-word Filtering**: If the NE identified in the given query text contains only stop-words, we ignore that NE. We use the standard JMLR stop-word list.

- **Twitter POS Filtering**: The query text is Part-Of-Speech (POS) tagged with a tweet POS tagger. NEs that do not contain at least one word with POS tag as NN (indicating noun) are ignored.

**Disambiguation**

We adapt the disambiguation function from the TAGME system, after making a few changes for efficient computation.

We define the relatedness between two pages $p_a$ and $p_b$ as shown in Eq. 5.4.

\[
rel(p_a, p_b) = 1 - \delta(p_a, p_b)
\]  

When $p_a$ and $p_b$ are identical pages, $\delta(p_a, p_b)=0$ and hence relatedness score $rel(p_a, p_b)=1$. Otherwise $rel(p_a, p_b)$ is a score between 0 and 1.

Though TAGME and Milne & Witten system use $\delta$ to measure relatedness between two pages, we used $1 - \delta$ in our system for the following reason.

The overall vote given to a candidate page $p_a$ of NE $a$, by NE $b$ is defined as shown in Eq. 5.5.

\[
vote_b(p_a) = \sum_{p_b \in Pg(b)} rel(p_b, p_a) \times Pr(p_b|b)
\]

where $Pg(b)$ are all possible candidate pages for the NE $b$ and $Pr(p_b|b)$ is the prior probability of the NE $b$ linking to a page $p_b$. In the TAGME system, the overall vote is normalized by the total number of senses of $b$, i.e., $|Pg(b)|$. We experiment with both the normalized and the non-normalized version.

The total relatedness score given to a candidate page $p_a$ for a given NE $a$ is calculated as the sum of votes from all other NEs in the input text $T$, denoted as $A_T$.

\[
rel_a(p_a) = \sum_{b \in A_T \backslash \{a\}} vote_b(p_a)
\]

The overall score given to a candidate entity consists of the relatedness score $rel_a(p_a)$ and the prior probability $Pr(p_a|a)$ of the candidate page $p_a$. These two factors are combined to get the overall in the following two ways.

1. **Linear Combination**

\[
score_a(p_a) = \alpha \times rel_a(p_a) + (1 - \alpha) \times Pr(p_a|a)
\]

The value of $\alpha$ was determined experimentally as $\alpha = 0.83$. The anchor $a$ is then annotated with the Wikipedia page with the highest $score_a(p_a)$.

[8](http://jmlr.org/papers/volume5/lewis04a/all-smart-stop-list/english.stop)
2. **Threshold Combination** Among all the pages that link to anchor $a$ (i.e., $Pg(a)$) choose the page that has the highest relatedness $rel_a(p_a)$ denoted as $rel_{best}(p_a)$. The set of the other pages in $Pg(a)$, that yield $rel_a(p_a)$ varying less than 25% with respect to $rel_{best}(p_a)$ is determined. From this set, the page $p$ with the highest value of prior probability ($Pr(p|a)$) is chosen as the page for annotating the anchor $a$.

**Anchor Pruning** The disambiguation phase produces a set of candidate annotations, one per anchor detected in the input text $T$. This set has to be pruned in order to possibly discard the meaningless annotations. This removal is done using a scoring function similar to the one in TAGME [?].

It uses two features: the link probability $lp(a)$ of the anchor $a$ and the coherence between the candidate annotation of anchor $a \mapsto p_a$ and the candidate annotations of the other anchors in $T$. Coherence is defined as the average relatedness between the candidate sense $p_a$ and the candidate senses $p_b$ assigned to all other anchors $b$ in $T$.

$$\text{coherence}(a \mapsto p_a) = \frac{\sum_{p_b \in S \setminus \{p_a\}} rel(p_b, p_a)}{|S| - 1}$$

(5.8)

where $S$ is the set of distinct senses assigned to the anchors of the text $T$.

Pruning score $\rho$ combines the link probability and coherence by a linear combination as shown in Eq. 5.9

$$\rho(a \mapsto p_a) = \text{coherence}(a \mapsto p_a) + \gamma lp(a)$$

(5.9)

$\gamma$ was determined empirically. We observed that the best results are obtained when $\gamma$ is set to 0.1. The NEs with $\rho$ value less than threshold are pruned. Threshold value of $\rho$ can be empirically found to be 0.05.

### 5.7.1 Dataset

The dataset was made available in the ERD Challenge. The dataset was created by sampling 500 queries from a query log of a commercial search engine to form a development set and 500 queries for the test set. The average query length was 4 words per query. The dataset and KB can downloaded at http://web-ngram.research.microsoft.com/erd2014/Datasets.aspx. The KB is a snapshot of Freebase from 9/29/2013, keeping only those entities that have English Wikipedia pages associated with them. As the evaluation of the ERD challenge was restricted to these entities, we indexed this dataset and restricted our final results to these entities.

### 5.7.2 Results

In this section, we will explain in brief the ERD Challenge dataset and analysis of the results of our experiments. We submitted seven runs to the ERD Challenge. Each run consists of running our system with 500 search query strings. For each query, the system outputs the entity mentions and the
freebaseID (present in ERD dataset) they link to. We explain the seven runs in the chronological order adding/removing the features in that order.

**Run 1:** As the first run, this was run on a 100 query subset. This was the base system. For mention detection, Twitter POS filtering was used for mention filtering. For disambiguation, $vote$ (Eq. 5.5) was normalized and linear combination method was used. The anchor dictionary was indexed with multiple rows per anchor. This probably caused high latency leading to timeouts. The system had an F1 of 0.53.

**Run 2:** The Run 1 system used link probability instead of the prior probability when computing the overall score for disambiguation. In Run 2, the link probability was replaced by prior probability as shown in Eq. [5.7]. The system had an F1 of 0.50 (on 100 queries).

**Run 3:** From Run 3 onwards, the runs were evaluated on the full 500 query set. In mention detection, stopword filtering was used for mention filtering. In the disambiguation component, $vote$ (Eq. 5.5) was not normalized and threshold combination method was used. The anchor dictionary was re-indexed to have single row per anchor. The system had an F1 of 0.483.

**Run 4:** Run 3 system was enhanced with optimization when making database connections, to re-use the open database connections. In the disambiguation component, we replaced the combination method by the linear combination method. The system had an F1 of 0.472.

**Run 5:** In the mention detection component, Twitter POS filtering was used. The system regained the F1 as 0.483.

**Run 6:** Run 5 system was enhanced with corrections when computing the link probability. Eq. 5.9 was corrected to use link probability instead of prior probability. The system had an F1 of 0.44.

**Run 7:** Tested on the final test dataset. In the mention detection component, mention filtering was moved back to stopword mention filtering. The system had an F1 of 0.53.

### 5.7.3 Observations

We believe that the following could be the reasons for the low F1 score for the proposed entity linking system.

- **Twitter-POS gave worse performance than merely using the stopword list.** A tweet is a more self-contained sentence when compared to a search query. When typing a search query user relies on the search engine to understand the string. On the other hand, when typing a tweet, the user relies on human intelligence to understand. Accordingly he builds in more context in tweets, leading to the more self-contained sentences. Hence it is intuitive that Twitter-POS was expecting a more self-contained sentence, which the search queries were not. Hence the poor performance of Twitter-POS on search strings.

- **The entity linking system annotates only those entities that are present in the ERD Challenge dataset.** So if the system detects any entity other than this predefined subset of entities, it is not reported.
5.8 Conclusion

In conclusion, we presented three entity linking systems that accepts a document as input and links the relevant entity mentions to corresponding Wikipedia pages. Various syntactic as well as semantic features are utilized for segmenting and linking the entity mentions within the input document. We saw that with decreased context length and decreased (syntactic and semantic) structure of the documents, EL depends more on the knowledgebase to disambiguate the NE mentions and link them. Now we move on and see more of knowledgebase in the next two chapters of the thesis.
Knowledgebase enhancement for Entity Linking: ELDEN

Knowledge base (KB) stores facts about entities. Presence of facts about an entity in the KB provides structure and guidance to the extraction task. The extraction task utilizes the stored facts of the entity to refine the extraction process. The facts thus extracted are in turn stored in the KB, making the KB bigger and better. Thus the task of entity extraction and knowledge base enhancement are mutually dependent and mutually supporting. Having seen extraction process in the last two chapters, in this and next chapter we present experiments on how to enhance the KB.

We present two aspects of KB enhancement namely ‘KB enhanced for EL’ and ‘KB enhanced by EL’. In this chapter we look at the first aspect which is KB densification helping EL. The advantage of KB densification in EL is realized more pronouncedly when the entity to be linked is not well connected to other entities in the KB. We present the Entity Linking by KG DENsification (ELDEN) system which address this problem. ELDEN is a simple, quick and efficient EL method that yields improved results for tail entities. After linking the entities, we update the KB with metadata from the linking process. This enhances the prior linking belief of the KB. The next round of EL in turn depends on this prior linking belief (stored) for both detection and disambiguation of entities. We present experiments on the second aspect of KB densification in the next chapter.

6.1 Tail Entity Linking: ELDEN

While conventional EL systems perform well on popular NEs, their performance is poor in less popular or less known entities, a.k.a. tail entities. We discussed the challenge of linking tail entities in ‘C4 Poor performance of Entity Linking on tail entities’ in Section 1.4. We present a knowledgebase densification based solution to the problem which has out-performed state-of-the-art methods.

ELDEN: Definitions and Problem Formulation

We begin our discussion on ELDEN by presenting few definitions and formulating the EL problem.

Knowledge Graph (KG): A Knowledge Graph is defined as $G = (E, F)$ with entities $E$ as nodes and $F$ as edges. In allegiance to EL literature and baselines [? ? ], we use the Wikipedia hyperlink graph as the KG in this paper, where nodes correspond to Wikipedia articles and edges are incoming links from
one Wikipedia article to another. ELDEN ultimately uses a densified version of this Wikipedia KG, as described in Section 6.3.

**Tail Entities:** Following [?], we define an entity to be a tail entity if the number of edges incident on the node corresponding to the entity in the KG is less than threshold $\eta$. Otherwise, the entity is called a head entity.

As a slightly orthogonal definition, [? ? ?] define tail entities as entities not present in the KG. However, EL cannot be performed for an entity that is not present in KG. Hence, we define tail entities using threshold $\eta$ as described above.

**Pseudo Entities:** Pseudo entities are frequently occurring phrases in Wikipedia (Section 6.3) whose unambiguous mentions co-occur with mentions of KG entities in a text corpus.

**Entity Linking (EL):** Given a set of mentions $M_D = \{m_1, ..., m_n\}$ in a document $D$, and a knowledge graph $G = (E, F)$, the problem of entity linking is to find the assignment $\Lambda : M_D \rightarrow E_D$, where $E_D$ is the set of entities linked to mentions in document $D$ such that $E_D \subseteq E$.

For mention $m_i \in M_D$, let the set of possible entities it can link to (candidate entities) be $C_i$. Then, the solution to the EL problem is an assignment $\Lambda$ where,

$$\Lambda(m_i) = \arg\max_{e \in C_i} [\phi(m_i, e) + \beta \cdot \psi(e, E_D)]$$  \hspace{1cm} (6.1)

Here, $\phi(m_i, e) \in [0, 1]$ measures the contextual compatibility of mention $m_i$ and entity $e$. $\phi(m_i, e)$ is obtained by combining prior probability and context similarity. $\psi(e, E_D)$ measures the coherence of $e$ with other entities ($E_D$) in the assignment in document $D$. $\beta$ is a variable controlling inclusion of $\psi$ in the assignment $\Lambda$.

### 6.2 Background

[?] is a recently proposed state-of-the-art EL system. We consider it as a representative EL system and use it as the main baseline for the experiments in this paper. In this section, we briefly describe [?]’s approach. [?] solves the EL problem presented in Section 6.1 in two steps.

**Step 1:** A mention $m_i \in M_D$ is defined to be *unambiguous* if $\exists e \in C_i$ such that $\phi(m_i, e) \geq \gamma$. Let $M_D^{(u)} \subseteq M_D$ be the set of such unambiguous mentions in document $D$. [?] freezes the assignment of all such unambiguous mentions $m \in M_D^{(u)}$ by solving Equation 6.1 for such mentions after setting $\beta = 0$. In other words, coherence of $e$ with other candidate entities is not used while assigning entities to unambiguous mentions. Assigning entities first to unambiguous mentions have also been found to be helpful in prior research [? ?]. Let $A_D$ be the set of entities linked to in this step. In Figure 3.2, mention *Andrei Broder* is unambiguous.

---

1 We set $\eta = 500$ for the experiments in this paper.
2 Creating new KG entities in such cases is beyond the scope of this work.
3 Between mention *Andrei Broder* and entity *Andrei Broder*, string similarity and prior probability are 1.0. In the experiments we use a $\gamma$ value of 0.95.
Embedding similarity = $\psi_{\text{ELDEN}} = \text{Cosine}(v_{\text{Entity1}}, v_{\text{Entity2}})$

$\text{Cosine}(v_{\text{Andrei_Broder}}, v_{\text{WWW_conference}}) > \text{Cosine}(v_{\text{Andrei_Broder}}, v_{\text{World_Wide_Web}})$

Figure 6.1: ELDEN approach consists of KG densification, training entity embeddings on densified KG and building EL system that uses similarity between the trained embeddings as coherence measure. ELDEN’s difference from baseline method is that while baseline method uses input KG for training $V_e$, ELDEN uses densified KG for training $V_e$.

**Step 2**: In this step, $[?]$ links all ambiguous mentions by solving Equation 6.2

$$\Lambda(m_i) = \arg\max_{e \in C_i} \left( \phi(m_i, e) + \frac{1}{|A_D|} \sum_{e_j \in A_D} v_e \cdot v_{e_j} \right)$$

$$\forall m_i \in M_D \setminus M_D^{(u)} \quad (6.2)$$

where $v_e, v_{e_j} \in \mathbb{R}^d$ are $d$-dimensional embeddings of entities $e$ and $e_j$ respectively. Please note that the equation above is a reformulation of Equation 6.1 with $\beta = 1$ and $\psi(e, E_D) = \frac{1}{|A_D|} \sum_{e_j \in A_D} v_e \cdot v_{e_j}$, where $A_D$ is derived from $E_D$ as described in Step 1.

Embeddings of entities are generated using word2vec model and trained using WLM $[?]$ (Details are explained in Section 6.3.1). Given a graph $G = (E, F)$, the WLM coherence measure $\psi_{\text{wlm}}(e_i, e_j)$ between two entities $e_i$ and $e_j$ is defined as follows.

$$\psi_{\text{wlm}}(e_i, e_j) = 1 - \frac{\log(\max(|C_{e_i}|, |C_{e_j}|)) - \log(|C_{e_i} \cap C_{e_j}|))}{\log(|E|) - \log(\min(|C_{e_i}|, |C_{e_j}|))} \quad (6.3)$$

where $C_e$ is the set of entities with edge to entity $e$.  

73
### Table 6.1: Notation used in KG densification and learning entity embeddings (Please see Sec 6.3 for more details).

<table>
<thead>
<tr>
<th>Notation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>( v_e )</td>
<td>entity embedding of entity ( e ) with dimension ( 1 \times d )</td>
</tr>
<tr>
<td>( E )</td>
<td>set of all titles in Wikipedia</td>
</tr>
<tr>
<td>( S )</td>
<td>set of all pseudo entities considered in ELDEN</td>
</tr>
<tr>
<td>( E_+ )</td>
<td>set of all entities considered in ELDEN</td>
</tr>
<tr>
<td>( V )</td>
<td>word vectors of ( E_+ ) where ( V \in \mathbb{R}^{k \times d} )</td>
</tr>
<tr>
<td>( V_e )</td>
<td>word vectors of ( E )</td>
</tr>
<tr>
<td>( \psi_{ELDEN} )</td>
<td>embedding similarity measured using ( V_e ) trained on ( G_{dense} )</td>
</tr>
<tr>
<td>( \psi_{Yamada} )</td>
<td>embedding similarity measured using ( V_e ) trained with input KG ( G )</td>
</tr>
</tbody>
</table>

### 6.3 Approach to Tail Entity Linking: ELDEN

In this section, we present ELDEN, our proposed approach. ELDEN extends [?], with one important difference: rather than working with the input KG directly, ELDEN first densifies the KG with co-occurrence statistics extracted from a large corpus. Even though this is a simple change, this results in improved EL performance, especially for tail entities.

#### Overview

Overview of the ELDEN system is shown in Figure 6.1. ELDEN starts off with densification of the input KG, using statistics from text corpus. This is meant to overcome edge sparsity involving entities not well connected in the KG. This step is described in Section 6.3. Embeddings of entities are then learned utilizing the densified KG in the next step. This step is described in Section 6.3. Embedding similarity estimated using the learned entity embeddings is used in calculating coherence measure in subsequent entity linking. Notations used is summarized in Table 6.1.

#### Knowledge Graph Densification

Figure 6.1 depicts densification of KG in ‘Input KG’ and ‘Densified KG’. It shows two Wikipedia titles Andrei_Broder and WWW_conference from our running example (Figure 3.2). There are no edges common between Andrei_Broder and WWW_conference. In a web corpus, mentions of Andrei_Broder and WWW_conference co-occur with mentions of Program committee and it has a positive PMI value with both the entities. So ELDEN adds an edge from Program committee to the entities in the KG. Here Program committee is a pseudo entity in the ELDEN KG. Thus, ELDEN densifies the KG by adding edges from pseudo entities when the mentions of Wikipedia entity and pseudo entity co-occur in a web corpus and the pseudo entity has a positive PMI value with given entity.

Taking a closer look, KG densification process starts from ‘input KG’ which is Wikipedia hyperlink graph \( G = (E, F) \), where the nodes are Wikipedia titles \( (E) \) and edges are hyperlinks \( (F) \). ELDEN processes Wikipedia text corpus and identifies phrases (unigrams and bi-grams) that occur frequently...
Table 6.2: EL Features

Features used by various EL systems. Context compatibility $\phi(m_i, e_i)$ and coherence $\psi(e_i, e_j)$ of Equation 6.1 are parameterized using features as shown above. Context compatibility features are used by all systems. Coherence features used by Yamada16 are $\psi_{wlm}$ and $\psi_{Yamada}$. ELDEN uses $\psi_{ELDEN++}$ as coherence feature.

---

(more than 10 times) in it. We denote these phrases as *pseudo entities* ($S$) and add them as nodes to the KG. Let $E_+ = E \cup S$ be the resulting set of nodes.

While pseudo entities are not titles in Wikipedia, their mentions frequently co-occur with mentions of titles in Wikipedia. Hence, we refer to them as ‘psuedo’ entities.

In order to address edge sparsity of $G$ i.e., connect entities in $E_+$ to entities in $E$, ELDEN processes a web text corpus looking for mentions of entities in $E_+$. ELDEN uses Equation 6.1 with $\beta = 0$ to entity link the text corpus with the KG $G' = (E_+, F)$. In other words, only mention-entity similarity $\phi(m, e)$ is used during this linking. Based on this entity linked corpus, a co-occurrence matrix $M$ of size $|E_+| \times |E_+|$ is constructed. Rows and columns of $M$ correspond to entities (nodes) $E_+$ of the node-augmented KG $G'$. Each cell $M_{i,j}$ is set to the Pointwise Mutual Information (PMI) $\text{PMI}(e, e') = \log \frac{f(e, e') \times N}{f(e) \times f(e')}$ where $f(e)$ is the frequency of entity $e$ in web corpus, $f(e, e')$ is the sentence-constrained pair frequency of the entity pair $(e, e')$ in web corpus, and $N = \sum_{e, e' \in E_+} f(e, e')$. Please note that PMI, and there by $M$, are symmetric.

An expanded set of edges, $F_+$, is now defined as

$$F_+ = F \cup \{(e, e'), (e', e) \mid e' \in E_+, e \in E, M_{e,e'} > 0\}$$

In other words, we augment the set of initial edges $F$ with additional edges connecting entities in $E_+$ with entities in $E$ such that PMI between the entities is positive.

---

4Since prior probabilities of pseudo entities are not available, only mention-entity similarity component of $\phi(m, s)$ is used while linking a mention $m$ to a pseudo entity $s \in S$. 4
ELDEN now constructs the KG $G^\text{dense} = (E_+, F_+)$, which is a densified version of the input KG $G = (E, F)$. ELDEN uses this densified $G^\text{dense}$ for subsequent processing and entity linking.

Learning Embeddings of Densified KG Entities

Following [? ], ELDEN computes embeddings of KG entities using Word2vec [? ]. However, instead of calculating embeddings over the input KG, ELDEN learns embeddings of entities in the densified KG $G^\text{dense}$ (Section 6.3).

ELDEN derives entity embeddings using the same setup and corpus as in [? ], the Word2vec [? ] skip-gram with negative sampling model. Let $V$ be the word2vec matrix containing embeddings of entities in $E_+$ where $V \in \mathbb{R}^{k \times d}$. $v_e$ is the embedding of entity $e$ in $E_+$ with dimension $1 \times d$. As $E_+$ consist of $E$ and $S$, $V$ consists of embeddings of $E$ ($V_e$) and embeddings of $S$.

In word2vec model, entities in context are used to predict the target entity. ELDEN maximizes the objective function [? ] of word2vec skip-gram model with negative sampling, $L = \sum_{(t,c) \in P} L_{t,c}$ where

$$L_{t,c} = \log \theta(v_c \cdot v_t) + \sum_{n \in N_{(t,c)}} \log \theta(-v_n \cdot v_t)$$

Here $v_t$ and $v_c$ are the entity embeddings of target entity $t$ and context entity $c$. $P$ is the set of target-context entity pairs considered by the model. $N_{(t,c)}$ is a set of randomly sampled entities used as negative samples with pair $(t, c)$. This objective is maximized with respect to variables $v_t$’s and $v_c$’s, where $\theta(x) = \frac{1}{1+e^{-x}}$.

$P$ and $N$ are derived using $G^\text{dense}$. $t$ and $c$ are entities in $E_+$ such that $c$ share a common edge with $t$. $v_n$ is randomly sampled from $V_e$, for entities that do not share a common edge with $t$. Entity embedding similarity measured using $V_e$ trained this way on $G^\text{dense}$ is $\psi_{ELDEN}$. Embeddings of $S$ are trained using positive and negative word contexts derived using context length.

Bringing it All Together: ELDEN

ELDEN is a supervised EL system which uses two sets of features: (1) contextual compatibility $\phi(m, e)$; and (2) coherence $\psi(e_i, e_j)$. Instead of computing the coherence measure directly on the input KG, ELDEN computes this feature over $G^\text{dense}$, as described in previous sections. These features are summarized in Table 6.2. Following success of ensemble learning in EL [? ], ELDEN uses Random Forest ensemble learner [? ] to estimate model parameters.

6.3.1 Experiments

In this section, we evaluate the following:

- Is ELDEN’s corpus co-occurrence statistics-based densification helpful in overall linking entities? (Sec. 6.3.2)

- How does ELDEN’s KG densification aid in linking tail entities? (Sec. 6.3.2)
<table>
<thead>
<tr>
<th>Specifics</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of titles after cleaning (</td>
<td>E</td>
</tr>
<tr>
<td>Number of pseudo entities (</td>
<td>S</td>
</tr>
<tr>
<td>Number of epochs</td>
<td>3</td>
</tr>
<tr>
<td>Learning rate</td>
<td>0.25 linearly reducing to 0.01</td>
</tr>
<tr>
<td>Number of negative samples (</td>
<td>N</td>
</tr>
<tr>
<td>Context window size</td>
<td>3</td>
</tr>
<tr>
<td>Dimensions of embedding ((d))</td>
<td>100</td>
</tr>
<tr>
<td>Training time</td>
<td>7 days on gpu with 2 cores</td>
</tr>
</tbody>
</table>

Table 6.3: Parameters used in Wikipedia processing and training KG. (Please see Sec. 6.3.1)

Setup ELDEN entity linker is implemented using Random Forest ensemble\(^5\). Parameter values were set using CoNLL development set. Feature limit of 3 with number of estimators as 100 yielded best performance.

Knowledge Graph: Wikipedia Following prior EL literature, we use Wikipedia hypergraph as our KG [? ? ]. This KG is enhanced with pseudo entities as explained in Section 6.3. We process the Wikipedia corpus following the same procedure as [? ]. We cleaned Wikipedia dump (dated Nov 2015). We then parsed the Wikipedia article text for frequently occurring (more than 10 times) words and phrases denoted as pseudo entities. Thus ELDEN uses 5.9 million entities (i.e. \(|E_+| = 5.9M\)). More details on KG and parameters used for training embeddings are in Table 6.3.

Preprocessing: Web corpus and Densified KG For our experiments, we created a web corpus by querying Google\(^8\) for entities in \(E_+\). Top \(K\) search results are considered for unigram and bigram frequencies. This corpus occupied 6.8GB for entities mentioned in TAC and CoNLL datasets (54336 entities) and even for tail entities (edge count \(\leq 500\)) average crawl size is 670 lines or more (Please see Figure 6.5 of Section 6.3.3 for more details.) As [? ] also note, though tail entity is less popular in KB, it is not hard to find content about them on the web.

The web corpus is analyzed for mentions of entities in \(E_+\). Co-occurrence matrix \(M\) is created\(^9\) for mentions within co-occurrence window of size 10 for PMI calculation\(^10\). Edges are added from mention of pseudo entities with positive PMI to given entity. In experiments we add edges from top 10 pseudo entity mentions ordered by PMI values\(^11\).

Dataset In line with prior work on EL, we test the performance of ELDEN on CoNLL [? ] and TAC2010 [? ] datasets. These datasets consist of documents where mentions are marked and entity to which the mention links to, is specified. We use only mentions that link to a valid Wikipedia title (non NIL entities) and report performance on test set. Some aspects of these datasets relevant to our experiments

\(^5\)http://scikit-learn.org
\(^6\)by removing disambiguation, navigation, maintenance and discussion pages.
\(^7\)word tokens are cleaned for repetitions and non-character patterns
\(^8\)https://www.google.com/
\(^9\)This co-occurrence matrix is downloadable with source code.
\(^10\)We experimented with window sizes 10, 25 and 50. We chose 10 that gave best results
\(^11\)This is a tunable parameter.
Table 6.4: Performance comparison with other recent EL approaches. ELDEN matches best results in CoNLL and outperforms the state-of-the-art in TAC dataset. (Please see Section 6.3.2 for details and \psi ELDEN++ row of Table 6.6 for ELDEN results.)

<table>
<thead>
<tr>
<th>Method</th>
<th>CoNLL (P-micro)</th>
<th>CoNLL (P-macro)</th>
<th>TAC2010 (P-micro)</th>
</tr>
</thead>
<tbody>
<tr>
<td>[?]</td>
<td>82.5</td>
<td>81.7</td>
<td>-</td>
</tr>
<tr>
<td>[?]</td>
<td>85.6</td>
<td>84.0</td>
<td>81.0</td>
</tr>
<tr>
<td>[?]</td>
<td>88.32</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>[?]</td>
<td>88.7</td>
<td>-</td>
<td>80.7</td>
</tr>
<tr>
<td>[?]</td>
<td>91.8</td>
<td>89.9</td>
<td>-</td>
</tr>
<tr>
<td>[?]</td>
<td>91.7</td>
<td>-</td>
<td>87.2</td>
</tr>
<tr>
<td>[?]</td>
<td>93.1</td>
<td>92.6</td>
<td>85.2</td>
</tr>
<tr>
<td>ELDEN</td>
<td>93.0</td>
<td>93.7</td>
<td>89.6</td>
</tr>
</tbody>
</table>

are provided below.

**CoNLL:** In CoNLL dataset [? ], we use 27,816 mentions and report Precision of topmost candidate entity, aggregated over all mentions (P-micro) and aggregated over all documents (P-macro), i.e., if \( tp_i \), \( fp_i \) and \( p_i \) are the individual true positives, false positives and precision for each document in a dataset of \( \delta \) documents, then

\[
P_{\text{micro}} = \frac{\sum_{i=1}^{\delta} tp_i}{\sum_{i=1}^{\delta} tp_i + \sum_{i=1}^{\delta} fp_i}
\]

and

\[
P_{\text{macro}} = \frac{\sum_{i=1}^{\delta} p_i}{\delta}.
\]

For candidate entities, we use [? ] dataset12.

**TAC:** In TAC2010 [? ] dataset, we report P-micro of top-ranked candidate entity on 1,020 mentions. P-macro is not applicable to TAC as most documents have only one mention. For candidate entities, we index the Wikipedia word tokens and titles (as explained Section 6.3.1) using solr13. We index terms in (1) title of the entity, (2) title of another entity redirecting to the entity, and (3) names of anchors that point to the entity, in line with baselines. We are making this candidate set for TAC dataset publicly available, in line with candidates of CoNLL dataset by [? ].

**Baseline: Yamada16**

We use the Yamada et al. system explained in Section 6.2 as baseline. Entity embedding distance measured using \( V_e \) trained on the input KG \( G \) is \( \psi_{\text{Yamada}} \).

### 6.3.2 Results

**Overall EL Performance Comparison**

In Table 6.4, we compare ELDEN’s EL performance with results of other recently proposed state-of-the-art EL methods that use coherence models.

---

12https://github.com/masha-p/PPRforNED
13http://lucene.apache.org/solr/
Table 6.5: Percentage of tail entities in evaluation datasets. TAC has higher composition of tail entities than CoNLL. Hence, ELDEN has better results in TAC over CoNLL (Please see Table 6.4).

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Tail Entities</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Train</td>
</tr>
<tr>
<td>TAC</td>
<td>78.8</td>
</tr>
<tr>
<td>CoNLL</td>
<td>48.4</td>
</tr>
</tbody>
</table>

We see that ELDEN results matches best results on CoNLL and outperforms state-of-the-art in TAC dataset. While most EL systems give higher precision on CoNLL dataset than TAC dataset, ELDEN performs with high precision on TAC dataset too.

This is explained by analyzing distribution of tail entities in TAC and CoNLL datasets as presented in Table 6.5. We see that CoNLL test set has almost half as head and half as tail entities, whereas in TAC test set, 63.6% are tail entities. This higher constitution of tail entities in TAC, explains ELDEN’s better results in TAC relative to CoNLL dataset. We recommend using ELDEN in KGs with higher number of tail entities. As the number of tail-entities is more than the number of head entities in most KGs \[^{?}\], our method is expected to be of significance for most KGs.

EL Performance of Tail Entities We now analyze the results of using densified KG on EL of tail entities. We look at EL performance on tail entities in TAC and CoNLL datasets in the table in Figure 6.2. Table shows P-micro measured with coherence measures $\psi_{\text{Yamada}}$ and $\psi_{\text{ELDEN}}$ in combination with base and string similarity feature groups ($\phi$). This is measured across edge count in range of 100s. Top rows
correspond to tail entities with popularity increasing as we move down to lower rows. We see that precision from $\psi_{E LDEN}$ is higher than that of $\psi_{Y amada}$ for tail entities. Similar trends are seen on TAC and CoNLL datasets. In the individual edge ranges, we find the absolute increase in P-micro of $\psi_{E LDEN}$ over $\psi_{Y amada}$, to be higher in TAC than the CoNLL dataset. For head entities, we see that $\psi_{E LDEN}$ doing slightly better than $\psi_{Y amada}$. However the difference in P-micro values of head entities is lower compared to the difference in P-micro values of tail entities. In essence, ELDEN performs with higher precision in linking tail entities. So ELDEN is recommended for KBs with relatively higher composition of tail entities.

Ablation Analysis Performance of ELDEN and baseline methods using various feature set combinations on evaluation datasets is presented in Table 6.6. We conduct ablation analysis on our method. Starting with base features, we add various features to ELDEN incrementally and report their impact on performance. The results when using base feature group alone, and base and string similarity groups together ($\phi$) are presented in first and second rows for each dataset. We compare $\psi_{E LDEN}$ to three coherence measures: $\psi_{wlm}$, $\psi_{Y amada}$ and $\psi_{d ense}$, details of which were provided in Table 6.2. The performance improvement from each of the four coherence measures are in the next four rows. Performance of ELDEN from using all four coherence features is given in $\psi_{E LDEN++}$ row.

Table 6.6: Ablation analysis involving various coherence measures (see Table 6.2 for definitions of these measures). Statistically significant improvements over $\phi$ are marked with an asterisk. ELDEN’s coherence measure, $\psi_{E LDEN+ +}$, achieves the best overall performance. (Please see Section 6.3.2 for details).
On CoNLL dataset, $\psi_{\text{dense}}$ combined with $\phi$, gave an improvement of 2.0 and 1.9 (P-micro and P-macro) over Yamada16 results (90.0 and 91.1 respectively). We note that Yamada16 results are from our re-implementation of [?] system$^{14}$ In the baseline work, Yamada et al. report this to be 91.4 and 92.1 respectively. Thus we are able to almost reproduce the baseline results. We also present the results combining baselines $\psi_{\text{Yamada}}$ and $\psi_{\text{wlm}}$ versus $\psi_{\text{ELDEN}}$ and $\psi_{\text{dense}}$. We find the features with KG densification perform better than baselines. On TAC dataset also, combined with $\phi$, $\psi_{\text{dense}}$ is found to do better than $\psi_{\text{wlm}}$ and $\psi_{\text{ELDEN}}$ gives a significant P-micro improvement of 4.2 over $\psi_{\text{Yamada}}$. The $\psi_{\text{ELDEN++}}$ P-micro in TAC dataset is statistically significant$^{15}$ In short, we find $\psi_{\text{dense}}$ measure to perform better than $\psi_{\text{wlm}}$ and $\psi_{\text{ELDEN}}$ to perform better than $\psi_{\text{Yamada}}$ on both datasets.

The goal of this paper is to create an EL system, that links head and tail entities alike. The results show that we are able to achieve state-of-the-art performance using densified KG. Thus, using corpus entity mentions refined with statistical measure (PMI), we can improve EL performance even when inter-entity edges are less for the entity in KG. **Error Analysis** We present an analysis of errors made by

<table>
<thead>
<tr>
<th>Error Class</th>
<th>Example</th>
<th>Referent entity</th>
<th>Predicted entity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acronym</td>
<td>I commute from Blue Ash (near Amberly) to the VA Medical Center (near UC). In the morning I shoot down Blue Ash Rd to Montgomery Rd.</td>
<td>University of Cincinnati</td>
<td>University of California</td>
</tr>
<tr>
<td>Synonym</td>
<td>a federal study that cites rising tobacco use and higher costs for the Pentagon and Department of Veterans Affairs as reasons for the ban. The study by the Institute of Medicine, requested by the VA and Pentagon, calls for a phased-in ban over a period of years, perhaps up to 20.</td>
<td>United States Department of Veterans Affairs</td>
<td>Virginia</td>
</tr>
<tr>
<td>Specific label</td>
<td>each year, the oil firm would owe just over $195,000. estimated 4.4 million tons of carbon dioxide emitted (based on Others in the top 10 include the Chevron refinery in Richmond, a power plant in Pittsburg and a cement company in Cupertino.)</td>
<td>Richmond, California</td>
<td>Richmond, Virginia</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>MILAN, Italy has not been able to overcome Les Bleus since, losing 3-1 in Italy beat France on penalties to win last year's World Cup, but Paris and drawing 0-0 in Milan in qualifying for Euro 2008.</td>
<td>Milan (city in Italy)</td>
<td>MILAN (the football team from Milan in Italy)</td>
</tr>
</tbody>
</table>

Table 6.7: Categorizing error made by ELDEN. Examples from TAC dataset. Better modelling of contextual entities and coherence of contextual entities can help reduce ‘Specific label’ errors (Please see Sec 6.3.2)

ELDEN on TAC dataset, before and after applying KG densification. We categorize the errors into four

---

$^{14}$Since Yamada et al source code is not publicly available, we have re-implemented the system using hyper-parameters specified in the paper and these are our best-effort results.

$^{15}$ with 2-tail 95% value of 1.96.
classes in Table 6.7, in line with error classes of \[?\]. An error is in “Synonym” class when mention is an alternate name of referent entity and “Specific label” class if referent entity is a specific instance of the mention. An error is in “Acronym” class when mention is an acronym and referent entity is expansion. All other errors are accounted in “Miscellaneous” class. We manually analyzed 240 wrong

![False Positive Analysis](image)

Figure 6.3: False positives of ELDEN before and after KG densification. Errors reduce with use of $\psi_{\text{dense}}$ and $\psi_{\text{ELDEN}}$ measures. (Please see Sec 6.3.2)

predictions of ELDEN and the results are presented in Figure 6.3. We found errors to reduce with use of $\psi_{\text{dense}}$ and $\psi_{\text{ELDEN}}$ features. Half of the errors eliminated is in “Specific label” class. Errors in this class called for better modeling of mention’s context. So we infer that $\psi_{\text{dense}}$ and $\psi_{\text{ELDEN}}$ does better detection of coherence between mentioned entities.

6.3.3 Observations

In this section, we share some insights into ELDEN, analyze the model choices and discuss cases where it does not improve EL performance.

*Insights into the efficacy of ELDEN approach:* In order to understand why ELDEN works, we look at why the basically WLM works. WLM is modeled after Normalized Google Distance (NGD) \[?\]. NGD is a semantic similarity measure derived from the number of hits returned by Google search engine for a given set of terms. It is based on term occurrences on web-pages. Web-pages that contain both terms indicate relatedness. WLM is NGD adapted with edges. In WLM, an edge common to two entities in Wikipedia, indicate relatedness. We argue that $\psi_{\text{dense}}$ is also NGD, as it is WLM extended with edges
from pseudo entities that are common to two entities. As the additional edges are common to both the entities, they indicate relatedness. Further, in WLM, common edge is a Wikipedia page that contains both entities. In $\psi_{\text{dense}}$, common pseudo entity is a term that co-occurs with both the entities mention in a web corpus.

The NGD method uses number of counts returned by Google, which requires $|E_+|^2$ search queries. To create the co-occurrence matrix of Sec. 6.3.1 ELDEN uses the top K search results. This requires only $|E_+|$ search queries.

Model approximations: To create a co-occurrence matrix, we need an entity linked web corpus. However purpose of creating the co-occurrence matrix is to solve entity linking. This was a chicken-or-egg-first situation. We resolve it by using unambiguous mentions of entities i.e. pseudo entities. As discussed in Section 6.3 we link the pseudo entity mentions using string matching alone in the web corpus, i.e., if the string of mention matched 100% with entity name, we count the mention as a entity occurrence. While it is not the ideal entity linked corpus, it closely approximates one.

Size of data to be crawled: To understand if web corpus is a good source for enhancing EL performance on tail entities, we analyze the size of web corpus vs precision in Figure 6.4. We see that precision does not vary greatly when size is more than 500 lines. Figure 6.5 presents average number of lines of crawled web content across entities with different edge counts. We see for all entities (head as well as tail) the average crawl size is 670 lines or more. Thus web corpus proves to be a good source of additional links for densification, for both tail and head entities.

Figure 6.4: Average size (number of lines) of crawled content of an entity versus precision in EL. On an average, tail entities have more than 500 lines size. Hence Precision is comparable to that of entities with higher web corpus sizes.
Figure 6.5: Tail entities have average crawl size of 670 lines or more. Thus corpus proves to be a good source of additional links for densification, for both tail and head entities.

**PMI link availability:** One of the limitations in ELDEN’s KB densification method is that, if the edges from pseudo entities to tail entity is less or nil (in case when the other entity is also a tail entity), it results in no densification of edge graph. In some cases new edges are added from pseudo entities but the added edges are not common between the pair of entities. This also results in no densification.

**Textual context with embeddings:** Given the context words $W$ of a mention $m_i$, measure contextual compatibility of $m_i$ and entity $e$ as $\phi(m_i,e) = v_e \cdot \frac{1}{|W|} \sum_{w \in W} v_w$ where $\cdot$ is the dot product operator and $v_e$ and $v_w$ the embeddings of $e$ and $w$. In our experiments we did not find this measure to contribute positively to ELDEN performance. Hence we have not included this measure in ELDEN.

### 6.4 Conclusion

We started this study by analyzing the performance of state-of-the-art EL systems and found that the performance was low on tail entities. We see that this is due to graph sparsity around the tail entities. We propose a method to improve the number of KB connections of an entity using mentions of the entity in a web corpus. The proposed ELDEN system achieves this by densifying edge graph. In TAC2010 dataset which has a higher share of tail entities, the performance of ELDEN is the best. As ELDEN uses a corpus based measure, it is general, unsupervised and applicable to all KBs. It is recommended for KBs with many tail entities.
Chapter 7

Application of KB entities: Document categorization

The proof of the pudding is in the eating. We demonstrate an application of Named Entities in KB which is enhancing categorization of computer science research papers using knowledge bases. Automatic categorization of computer science research papers using just the abstracts, is a hard problem to solve. This is due to the short text length of the abstracts. Also, abstracts are a general discussion of the topic with few domain specific terms. These reasons make it hard to generate good representations of abstracts which in turn leads to poor categorization performance. To address this challenge, external Knowledge Bases (KB) like Wikipedia, Freebase etc. can be used to enrich the representations for abstracts, which can aid in the categorization task. In this chapter, we propose a novel method for enhancing classification performance of research papers into ACM computer science categories using knowledge extracted from related Wikipedia articles and Freebase entities. We use state-of-the-art representation learning methods for feature representation of documents, followed by learning to rank method for classification. Given the abstracts of research papers from the Citation Network Dataset containing 0.24M papers, our method of using KB, outperforms a baseline method and the state-of-the-art deep learning method in classification task by 13.25% and 5.41% respectively, in terms of accuracy. We have also open-sourced the implementation of the project.

7.1 Applying KB entities to improve document categorization task

One of the difficulties faced in categorization of the papers in the conference proceedings is automatically identifying the categories of the research papers from the standard ACM computing classification system. It is important to find the right category of the paper submitted by authors for several purposes, which includes sending the paper to a panel with relevant reviewers according to the category, publishing papers under the correct category and so on. Given the limited amount of content in the abstract and a very high level discussion of the topic with few domain specific terms, finding the category of the paper using just the abstract is a challenging and a hard problem.

In this paper we address this problem and propose a novel method to leverage external Knowledge Bases (KB) to improve the performance of short text categorization, using the learning to rank framework [? ].

We evaluate our method on a large dataset of abstracts with the aim of classifying the paper into one of the 24 ACM computer science categories. Our method outperforms the baseline method, which does not use any external information, by 13.25% and outperforms the existing state-of-the-art model in text categorization by 5.41%, in terms of accuracy.

7.2 Background

Traditional methods for text classification, work by representing document using human curated features like TF-IDF features, followed by a linear classifier like SVM [? ]. Due to the bag-of-words assumption and sparsity induced by high dimensionality in the TF-IDF feature vector, these methods do not perform very well. Another approach to this problem is using dimensionality reduction methods on the TF-IDF feature vector to overcome the sparsity problem. These methods include Latent Semantic Allocation (LSA) [? ] and Latent Dirichlet Allocation (LDA) [? ].

Recent advancements in distributional representations of text resulted in better representation schemes for the document. Some examples of such techniques are word2vec [? ], paragraph2vec [? ] and GloVe [? ]. Mikolov et al. [? ] demonstrated the superiority of distributed representation methods over classical representation methods in the sentiment analysis task.

The success of deep learning methods in the field of computer vision and speech processing, inspired their applications in Natural Language Processing (NLP) [? ]. Combined with the superior representation learning methods, these methods have proven to be state-of-the-art in a variety of NLP tasks like sentiment analysis [? ], document similarity task [? ], etc. For the text categorization problem, current state-of-the-art models are based on Convolutional Neural Networks (CNN) [? ].

Our main contribution lies in using the learning to rank framework to combine KB with the text using state-of-the-art representation learning methods for text.

7.2.1 Learning to Rank

In a typical setting, the learning to rank method is defined as follows. We are given a query \( q_i \in Q \) and a set of \( N \) candidate documents \( (d_{i1}, d_{i2}, \ldots, d_{iN}) \). For each document \( d_{ij} \), there is a binary relevance label \( y_{ij} \) such that \( y_{ij} \in \{0, 1\} \), where a label of 1 indicates that the candidate document is relevant and 0 otherwise. Given this information, the goal of learning to rank method, is to learn a function \( h \), that assigns a higher score to relevant documents than to the non-relevant documents. Formally, learning to rank tries to learn the function \( h \) defined as follows.

\[
h(w, \psi(q_i, d_{ij})) \rightarrow \mathbb{R}
\]
where \( w \) is the set of parameters for the function, and \( \psi \) generates a feature vector representation of the query and the document combined.

The Learning to rank framework has these two broad categories:

- **Pointwise Approach**, where the training instances are \((q_i, d_{ij}, y_{ij})\) and a binary classifier is trained over input pairs \((q_i, d_{ij})\) defined formally as: \( h(w, \psi(q_i, d_{ij})) \rightarrow y_{ij} \) with the goal to predict, whether the document \( d_{ij} \) is relevant to the query \( q_i \) or not.

- **Pairwise Approach** where the model is trained to score correct pairs higher than incorrect pairs with a fixed margin.

While it can be argued that pairwise models can give better results than pointwise models, the primary focus of this work is on generating a good combined representation for the abstract and the corresponding KB entity which can be used for classification, rather than capturing different aspects of similarity for ranking. Hence in this paper, we adopt pointwise method of the learning to rank framework.

### 7.3 Approach

Let \((w_{c1}, w_{c2}, ..., w_{cK})\) denote the set candidate KB entities corresponding to each of the \( K \) categories. Let \((y_1, y_2, ..., y_K)\) denote the set of \( K \) labels each corresponding to a category. For each paper (research article) \( d_p \) in the dataset, there is an associated value for each \( y_i \) such that \( y_i = 1 \) if \( d_p \) belongs to category \( c_i \) and 0 otherwise. For each paper \( d_p \), our goal is to rank KB entity relevant to its category \( w_{ci} \) higher in score than other KB entities.

Formally the model in Eq. 7.1 can be specified as follows.

\[
h(W, \psi(d_p, w_{ci})) = f(W \left[ \frac{R(d_p)}{R(w_{ci})} \right] + b) \tag{7.2}
\]

where \( R \) is the feature representation function (word2vec or paragraph2vec), \( w_{ci} \) is the KB entity corresponding to the category \( c_i \), \( W \in \mathbb{R}^{1 \times 2d} \) is the linear transformation matrix, \( d \) is the embeddings dimension and \( b \in \mathbb{R} \) is the bias parameter. We use sigmoid function to realize \( f \).

Our input consists of triplets \((d_p, w_{ci}, y_i)\), where \( y_i = 1 \) if \( w_{ci} \) is the KB entity corresponding to the category of the document \( d_p \) and 0 otherwise. Since we are training a discriminative classifier, for each positive pair \((d_p, w_{ci})\), we sample a negative pair \((d_p, w_{cj})\) with label 0. The model is trained to optimize the Binary Cross-Entropy (BCE) loss function defined as follows.

\[
BCE(y_i, h(W, \psi(d_p, w_{ci}))) = -y_i \log h(W, \psi(d_p, w_{ci})) + (1 - y_i) \log(1 - h(W, \psi(d_p, w_{ci}))) \tag{7.3}
\]

For testing on a new document, we generate the feature representation of the document and combine it with the feature representation of each category in accordance with Eq. 7.2 and select the category
for which a label of 1 is predicted. If for multiple categories, we get a label of 1, we randomly select one category out of all predicted categories.

We use word2vec and paragraph2vec for the feature vector representation $R$. To generate the feature vector representation using word2vec, we compute the average of the word vectors corresponding to each token in the document. To account for out-of-vocabulary words, a similar strategy described in [?] is followed and they are replaced with a randomly sampled vector of same dimension as a vector drawn from a uniform distribution $U[-0.25, 0.25]$. For the feature vector representation using paragraph2vec, we follow the inference mechanism as described in [?].

### 7.3.1 Dataset

We use the Citation Network Dataset [?] which contains 236565 papers, with each article categorized into one of the 24 ACM categories. We randomly split these articles into 80% training instances and 20% testing instances, and used a one-vs-rest logistic regression Eq.7.2 for classification. For external knowledge base, we select two popular knowledge bases: Wikipedia and Freebase (FB). We use pre-trained embeddings for freebase entities, trained on Google News Corpus to initialize each entity’s representation corresponding to each category in text. We experiment with word2vec and paragraph2vec to generate embeddings of categories with Wikipedia.

### 7.3.2 Experimental Setup

To generate document embeddings we use gensim [?], an open-source python library with embedding size set to 300. For text pre-processing, we remove all stop-words, punctuations and non-ASCII characters from abstracts of all articles. We then lower-case all text. To generate the TF-IDF representation of articles, we consider each article’s abstract as a document and collection of all abstracts as document collection, and term frequency and inverse-document-frequency is computed accordingly. We use a dimension of 300 for Freebase entity embedding. For more details about experimental settings, interested readers can refer to the open-source implementation code.

### Category to Entity Mapping

Knowledge-Base (KB) consists of entities and their relations. To generate semantic representation of the categories, we need to map them to their corresponding entities in the KB. Each category is mapped to its corresponding entity in KB by matching their corresponding description text. For example, the category ‘Machine Learning and Pattern Recognition’ is mapped to the entity ‘/en/machine_learning in Freebase and to the entity ‘Machine learning’ in Wikipedia. More details can be found in the open-source implementation of the project.

---


Table 7.1: Performance Comparison with Baseline Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy</th>
<th>Training Time (in secs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TF-IDF + SVM Baseline</td>
<td>59.30%</td>
<td>-</td>
</tr>
<tr>
<td>CNN baseline [?] with Avg. Padding</td>
<td>64.76%</td>
<td>-</td>
</tr>
<tr>
<td>CNN baseline [?] with Max. Padding</td>
<td>67.14%</td>
<td>259.446</td>
</tr>
<tr>
<td>Our model</td>
<td>72.55%</td>
<td>59.82</td>
</tr>
</tbody>
</table>

7.3.3 Results and Discussion

For comparisons, we select the state-of-the-art, Convolution Neural Network (CNN) based method for text categorization [?]. CNN used in the paper operates at a sentence level, so we concatenate all sentences in the abstract into a single sentence. The abstracts in our case are of variable length with different number of tokens, so there is a need to convert them to fixed length sequences. To achieve this, we employ two strategies: average and maximum length padding. In the case of average length padding, we consider the size of each abstract in the corpus and calculate the average length across the corpus. Each abstract in the dataset is then converted to the average size by either padding with a ‘null’ token or by truncating the sequence, depending on the length of the abstract relative to the average length. Similar strategy is employed with maximum length padding. For padding using the maximum document length method, we removed all documents in the training set with document length greater than 150. We use the code made available by authors[6] and run it on our dataset with the best settings reported in the paper.

We present the results of all the methods in Table 7.1. For evaluating the performance, micro-averaged accuracy is used as a measure. It is clear that our method outperforms the CNN baseline in accuracy by 5.4%. Since abstracts are short texts, adding external information to the model clearly gives us an advantage over current methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>Acc. without KB</th>
<th>Acc. with Wiki.</th>
<th>Acc. with FB</th>
<th>Acc. with Wiki. + FB</th>
</tr>
</thead>
<tbody>
<tr>
<td>para2vec</td>
<td>12.86%</td>
<td>68.04%</td>
<td>72.55%</td>
<td>72.54%</td>
</tr>
<tr>
<td>word2vec</td>
<td>50.95%</td>
<td>71.38%</td>
<td>71.51%</td>
<td>71.13%</td>
</tr>
</tbody>
</table>

Table 7.2: Performance Comparison with Different KBs

Quantitative Analysis We compare the results of our method with baseline methods that do not use KB in Table 7.2. Paragraph2vec and word2vec combined with Freebase gives an increase of 59.69% and 20.56% respectively in terms of accuracy over the baseline methods. It can also be observed that the performance of the method is consistent across different KBs. It can be inferred that the common factor which gives an increase in the performance is the knowledge from the KB, and not the type of KB itself. Further, we select two popular KBs with different semantics and one KB as combination of these two. The performance across these KBs provides us with evidence that the method can be generalized across

---

6https://github.com/harvardnlp/sent-conv-torch
different KBs. The two KBs differ in the sense that, Wikipedia contains mostly textual information about entities and topics, while Freebase has relation information between entities too and is different in nature than Wikipedia. We also present the training time comparison between our method and deep learning based state-of-the-art method in the rightmost column of the Table 7.1. The results are shown for 25 epochs of training for CNN (best setting reported by the authors) and 100 (determined using cross-validation) epochs of training for Logistic Regression. It is clear that our method is faster to train as compared to the CNN owing to mostly linear operations in our model, as compared to more complex feed-forward and back-propagation methods in CNNs.

7.4 Conclusion

This chapter proposed a novel method to combine information from external knowledge bases, using learning to rank framework to enhance classification performance in short texts like abstracts of research papers. We empirically demonstrate that our method outperforms deeper networks in terms of accuracy. Performance with different KBs was also studied, and it provides a strong evidence that the method can be generalized across different KBs. The only requirement is the presence of a high quality external KB. We also demonstrate that our method is faster to train than the baseline method.
Chapter 8

Entity Linking for Knowledgebase Enhancement: NEI

Our study of KB densification is divided into two aspects namely ‘KB enhanced for EL’ and ‘KB enhanced by EL’. In last chapter we saw the first aspect where KB densification helped EL. In this chapter we present experiments on the second aspect which is EL leading to KB densification. The scenarios of EL helping KB enhancement surfaces in cases where an EL system is unable to link a NE mention to any of the entities in the KB. This necessitates creation of a new entity in KB. We call this New Entity Identification (NEI), which we dealt with in this last chapter of the thesis. Identifying new entities leads to KB enhancement, which leads to KB enhancement.

8.1 New Entity Identification (NEI)

Having seen how enhanced KB helps improve EL, we now look at how EL helps in enhancing KB, by the process of New Entity Identification (NEI). We will start by defining the terms.

**Definition:** Entity Linking consists of three sub-tasks (i) Mention detection - detecting the linkable phrases i.e. phrase that qualifies as a link to an entity in the KB, called mentions. (ii) Disambiguation - identifying relevant entities from a KB. (iii) Linking - choosing the most suitable entity to link the mention. If the KB does not have an entry to link the mention, then the mention is generally referred to as NIL entity. We refer to this task of identifying NIL entities as **New Entity Identification (NEI).**

**Motivation:** NIL entities or sparsity in KBs has been recognized as an important issue in recent research [? ]. Detecting NIL entities is important also to avoid creating spurious links. In this section we study the New Entity Identification approaches of various Entity Linking systems and propose ways to improve standard EL systems to gain better NEI performance, based on learnings from our re-implementations. To the best of our knowledge this is the first study focusing on the NEI approaches of EL systems.
8.2 Approach to New Entity Identification

We approach the study of NEI by first analyzing existing NEI approaches in literature and then reproducing state-of-the-art approach and results. On reviewing the literature on NEI approaches (in Section 8.2.1), we found two types of NEI approaches. We call them Thresholding and NIL Classification in this thesis. After analyzing 31 NEI approaches, we have selected two representative algorithms from both NEI approach types, namely TAGME and the system proposed by Ploch for re-implementation. We chose these systems since they are seminal work, reported the best results and to gain insights in reproducing state-of-the-art approach and results. We discuss more about our selection in Section 8.2.2 which also explains experimental setup, datasets and evaluation of re-implementations (in Section 8.2.3). We draw from the re-implementations, the pros and cons of the two NEI approach types (in Section 8.2.4) and derive conclusions (in Section 8.3).

Learnings from re-implementations: From the analysis of our re-implementations, we find that the NEI performance of standard EL systems can be improved by the use of dedicated NIL entity classifiers which use word features. On EL systems using thresholding, mentions predicted as NIL with high linking confidence $\rho$, are good NIL entity candidates. Choosing a higher NIL threshold ($\tau$) value for the classifier helps in filtering out noise from NIL entities.

In short the takeaways from our study of NEI are

- We review and analyze the main approaches of NEI in EL systems and the features used.
- We re-implement representative algorithms from both NEI approach types to get insights from reproducing state-of-the-art results and identifying improvements.
- We have made all data-sets and software used in NEI study publicly available.

Terminology: In the literature on NEI, different terms are used by different authors to refer to entity mentions that do not have a referent entry in KB. While Bunescu and Pasca refer to an entity that is not covered in Wikipedia, as out-of-Wikipedia entity, Hoffart et al. call them Emerging Entities (EEs) or out-of-knowledge-base (OOKB) entities, Lin et al. refer to them as unlinkable-noun-phrase and Kulkarni et al. label it “NA” denoting no attachment. The popular TAC workshops refer to them as NIL entities, which we will use in this study.

8.2.1 Review of NEI Literature

Early research on NEI aimed at enhancing or maintaining automatically constructed KBs. Research focus on NEI (and EL in general) was enhanced by the Text Analysis Conference (TAC) workshop’s Knowledge Base Population (KBP) track which was aimed at discovery of information for inclusion in an existing KB. TAC workshop provided standard dataset and evaluation

---

1https://github.com/priyaradhakrishnan0/NEI
measures to compare EL and hence NEI tasks. In this section we look at the main NEI approaches in EL literature including the TAC KBP workshop. We also look at major features used for NEI.

**NEI approach types:** In an extensive survey of EL systems, Wang et al. [176] identify the main approaches to NEI. A simple heuristic approach is one where if the candidate entity set \( e \) generated for the mention \( m \) is empty, the EL system links \( m \) to NIL. This approach is implemented in [176]. The non-heuristic approaches are of two types.

**Thresholding** or NIL threshold method for NEI: Generally, EL systems do mention detection, disambiguation and linking sequentially. The EL system relies on confidence of the disambiguation sub-task for linking a mention to an entity. It learns a threshold value for this confidence called NIL threshold \( \tau \), from the training data. Based on \( \tau \) value, EL system could drop a mention (in a low-confidence situation [176]) by declaring it unlinkable or map it to a global NIL [176]. Many systems [176] use thresholding method to identify NIL entities. Systems like TAGME [176] and Jin et al. [176] also use a learned threshold to link the mention to NIL.

**Supervised machine learning techniques:** Many EL systems use supervised machine learning techniques to identify the NIL entity. Here the NEI task is, given a mention \( m \) and set \( t \) of candidate entities from the KB, identifying that \( m \) cannot be linked to any entity in \( t \). Based on machine learning technique used, Roth et al. [176] re-group these approaches into two. First one involves classification and local processing where mention \( m \) is assigned to NIL when \( m \) cannot be linked to any entity in \( t \). The systems [176] use a binary classifier for this. Second one involves clustering and global processing. All mentions \( m \) that represent the same entity are clustered. If the cluster cannot be linked to any entity in the KB, then it is mapped to NIL [176].

**NEI Features:** We now look at prominent features used for NEI.

**Word features:** Identifying words and phrases that identify the new entity is a prominent method to do NEI. In their work on ‘No-Noun Phrase’ identification, Lin et al. [176] devise a supervised classifier trained on temporal features of words to predict if a noun phrase contains an entity mention. They also predict the entity’s fine-grained type. Graus et al. [176] present an unsupervised method for generating pseudo-ground truth for training a named entity recognizer to specifically identify new entities that will be added to a KB. The approach of Ratinov et al. [176] is to initially rank the candidate entities by local features including keyphrases and later link to an entity or NIL. In their system Hoffart et al. [176] extract keyphrases from non-KB sources like new articles and explore the link-ability of the mention to the NIL entity with high precision.

**Presence in macroKB:** Another way of establishing the absence of an entity in a KB is by verifying its presence in a larger (or macro) KB. This technique is used in SIEL [176], Zhang [176] and MS_MLI [176] systems. As the TAC KB is based on the 2008 Wikipedia snapshot, a KB based on a later Wikipedia version functions as the macroKB in this case.

**NEI Approaches in TAC:** In this section we look at NEI approaches of EL systems that either participated in TAC or used the TAC KB dataset in their experiments. The TAC KBP track is being conducted

\[\text{93}\]
since 2009. We analyze the NEI approaches over the years in the context of approach types and features discussed so far.

Among TAC 2009 systems that report best NEI performance, thresholding is used by Dredze et al. They consider absence (i.e., the NIL candidate) as another entry to disambiguate and learn the \( \tau \). They use word similarity features as NEI features. Presence in macroKB feature is used by the top scoring system (SIEL), which computes similarity of query to KB entities and Wikipedia. If the query has no (or very small) similarity to KB entities and has high similarity to a single Wikipedia page, it infers that the likely link for the query is not present in KB, thus it is NEI. Li et al.’s system uses supervised machine learning for NIL entity detection. They first rank possible candidate entities and select the top-ranked option. Then they use a separate binary classifier to decide whether this top prediction is NIL.

Zhang et al. use the presence in macroKB feature for NEI. They use Wikipedia data curated from the 2009 snapshot of Wikipedia. If the linking entity was found in this Wikipedia data and not in TAC KB, it was declared a NIL mention. While they achieved a micro-averaged-accuracy of 0.83 for NIL entity, Ploch reported the highest micro-averaged accuracy for NIL entity using TAC KB and TAC 2010 dataset at 0.96. They approach disambiguation and NIL detection as supervised classification tasks and use two binary SVM classifiers. The first classifier decides for each candidate, if it corresponds to the target entity and second classifier detects NIL entities.

Use of supervised ML technique approach was needed from TAC KPB 2011, as participating systems had to cluster the NIL entity mentions i.e. when multiple mentions in a given document correspond to the same entity which is outside the KB, cluster the relevant mentions as representing a single NIL entity. Hierarchical clustering approach with multiple steps was adopted by the top team (LCC). It used a three-step process of grouping likely matches, clustering within those groups, and merging the final clusters. Evaluation of NIL clustering was done using B-Cubed F-measure and LCC system achieved a score of 0.86 on NIL entities. In TAC 2012, evaluation of NIL clustering was done using modified B-Cubed (B-Cubed+) metric and top team (B_CUNY) achieved a score of 0.789 on NIL entities. The B_CUNY system used collaborative clustering to achieve NIL clustering.

Using the presence in macroKB feature, TAC 2013 top team (MS_MLI) constructed a new KB by processing 2013 Wikipedia snapshot. Mentions disambiguated to entities in the new KB and having no corresponding entities in the TAC KB were labeled as NIL. In 2015, the task was extended from monolingual to trilingual, where EL systems were required to cluster mentions into NIL entities across languages. New datasets (trilingual) and new KB (based on Freebase) were used. In this thesis we have considered only mono-lingual (English) EL system’s NEI performance based on TAC KB 2009.
8.2.2 Re-implementating Best Result and State-of-the-art Approaches

We analyzed 31 NEI approaches in Section 8.2.1. Most of the approaches we discussed are evaluated on TAC. However all approaches (e.g., TAGME) were not evaluated on the TAC dataset. To have comparable results, we re-implemented the NEI-approaches. We chose NEI approaches based on: (i) Best results - We pick the approach that reports best result on common and/or comparable test conditions, (ii) State-of-the-art - We pick the work that is novel and widely cited, (iii) Ease of reproduction - Focus on issues raised only in reproduction.

<table>
<thead>
<tr>
<th>Approach</th>
<th>Thresholding</th>
<th>Supervised ML methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Representative Algorithm</td>
<td>TAGME [? ]</td>
<td>Ploch [? ]</td>
</tr>
<tr>
<td>Mention Detection</td>
<td>Inverted Index Lookup</td>
<td>Dictionary lookup</td>
</tr>
<tr>
<td>KB features</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Title</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Redirect</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Disambiguation</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Inlink</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Outlink</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Disambiguation Features</td>
<td>Relatedness score, Prior Probability</td>
<td>Context Similarity, BOW, Popularity</td>
</tr>
<tr>
<td>Linking</td>
<td>Coherence score</td>
<td>NIL classification</td>
</tr>
</tbody>
</table>

Table 8.1: NEI Approaches: We analyze and group the approaches into two groups, viz. thresholding and supervised machine learning. Tabulated above are representative approaches from the groups, compared against each other on five aspects. Please see Section 8.2.2 for more details.

We first analyze the approaches and group them (non-heuristic approaches) into two groups namely thresholding and supervised machine learning techniques. From the thresholding approaches group we picked TAGME [? ], a seminal work. From the supervised machine learning techniques group, we picked the system proposed by Ploch [? ] which reported best result on the TAC 2010 dataset. TAGME source code was not available to us initially [? ], prompting this re-implementation. Later it was made publicly available. Use of TAGME API did not suit our cause as API gives linking entities, whereas we are interested in the NIL entity task. We also could not obtain Ploch’s source code.

We tried to pick approaches with different set of features. However some amount of feature overlap was unavoidable as the features on KB similarity was used by almost all approaches. In this section, we
present a detailed analysis of the two re-implementations. These approaches have not been compared against each other before. An overview of the comparison is presented in Table 8.1.

**TAGME System**: TAGME was proposed by Ferragina et al. [?] for linking entities in short documents. Wikipedia inlinks are explored for detecting and linking the entities. Fig. 8.1 gives an overview of the TAGME system re-implementation.

![Figure 8.1: TAGME System : Representative of the Thresholding NEI approach. Mentions predicted as NIL with high linking confidence \( \rho \), were found to be good NIL entity candidates.](image)

**Mention Detector**: Mentions are detected using link probability \( lp(m) \), which is the probability that mention \( m \), is used as anchor in Wikipedia. Continuous word sequences of up to six words, are checked for their presence as an anchor in Wikipedia. If the \( lp(m) \) of the string is greater than a predefined threshold, then it is taken as a detected mention and all pages referred by it are taken as candidate entities \( e \). However this involves a large number of look-ups on Wikipedia anchor index. In order to reduce the number of look-ups, we used stopword filtering.\(^3\) We use the standard JMLR stopword list.\(^4\)

**Disambiguator**: The disambiguation is done based on the probability of the mention linking to a particular entity and the Wikipedia-based semantic relatedness measure \( (\delta) \). The disambiguator computes a score for each candidate \( e \), for each mention \( m \), based on agreement between the entities \( e \) of a mention with entities \( e \) of other mentions detected using \( \delta \). Linearly combining this score with the prior probability \( Pr(e|m) \), of \( e \), we get the disambiguation score of an entity.

**Linking**: The disambiguation phase produces one candidate entity \( e_m \), per mention \( m \) of the input text, \( T \). The average semantic relatedness between the candidate entity \( e_m \) and the candidates \( e_n \) assigned to all other anchors \( n \) in \( T \) is measured as Coherence. Linking combines coherence with link probability \( lp(m) \) of the mention \( m \) to arrive at linking confidence \( \rho \). The mentions with \( \rho \) value less than the threshold are NIL entities. Threshold value of \( \rho \) for NIL entities \( \tau \), is learned with an SVM classifier using the training dataset.

**Ploch System**: Ploch [?] approaches linking as a supervised binary classification problem using two binary SVM classifiers, one for entities present in KB and other for NIL entities. Fig 8.2 depicts our re-implementation of this system.

**Mention Detection and Disambiguation**: Mention detection is done by dictionary look-up. A KB is created by processing Wikipedia, mapping Wikipedia article name as entity name and mapping its

---

\(^3\) If the mention identified contains only stopwords, we ignore that mention.

\(^4\) [http://jmlr.org/papers/volume5/lewis04a/a11-smart-stop-list/english.stop](http://jmlr.org/papers/volume5/lewis04a/a11-smart-stop-list/english.stop)
surface forms, categories and context words. Mention detection uses this KB to generate candidate entities $e$. The disambiguation features include Entity Context, Link Context and Standard features (including Bag-Of-Words and tf.idf).

**Linker:** The first classifier (Candidate classifier) decides for each candidate $e_i$, if it corresponds to the target entity. Each candidate is represented as a vector $F$ of features. For training the classifier, we label at most one $e_i$ from $e$ as a positive example and all others as negative. A second classifier (NIL classifier) is trained to detect NIL queries. Positive samples are mentions that link to NIL and mentions that have similarity values of all candidates $e_i$ as very low. Other features implemented were the maximum, mean and minimum, the difference between maximum and mean, and the difference between maximum and minimum, for all atomic features, using the feature vectors of all candidates $e_i$. Both classifiers use a Radial Basis Function (RBF) kernel, with parameter settings of $C = 32$ and $\gamma = 8$.

**Experiments and Evaluation of Re-implementations**

**Dataset:** We use TAC dataset for evaluation. TAC evaluation dataset has queries and gold standard. The queries consist of a mention string and a source document containing it. The gold standard is a reference to a TAC KB node or NIL if there is no corresponding node in the KB. TAGME system uses training data to estimate the NIL threshold, $\tau$. Ploch system used the TAC 2009 data set for training. So we use TAC 2009 dataset as our development dataset. We use the TAC 2010 test data as our final held-out test set. Both the 2009 and 2010 test data set has approximately 55% NIL entities, which makes our final test data not very different in the constitution of NIL entities from the training data. TAC 2009 and 2010 datasets have highest number of NIL queries compared to other TAC datasets (2011, 2012, 2013) and the AIDA_EE GigaWord dataset [? ].

**Evaluation Measures:** Hachey et al. [?] define NIL Precision ($P_\emptyset$) and NIL Recall ($R_\emptyset$) as the evaluation measures for measuring NEI performance. Micro averaged Accuracy ($A_{micro}$), which is percentage of correctly linked queries, is the official TAC measure for evaluation of EL systems. TAC reports NIL accuracy ($A_\emptyset$) which is $R_\emptyset$ calculated with the system generated entity set having a single entity which is NIL entity. These are calculated as follows:

$$A_{micro} = \frac{\{S_{i,0}|S_{i,0} = G\}}{Q} \quad (8.1)$$

$$P_\emptyset = \frac{\{|S_i|S_i = \emptyset \land G_i = NIL\}}{\{|S_i|S_i = \emptyset\}} \quad (8.2)$$

---

Figure 8.2: Ploch System: Representative of Supervised ML Method NEI Approach. Choosing a higher NIL threshold ($\tau$) value for the classifier helps filtering noise from NIL entities.

[We used the libsvm implementation](https://www.csie.ntu.edu.tw/~cjlin/libsvm/)
\[
F = \frac{2P_\emptyset R_\emptyset}{P_\emptyset + R_\emptyset} \tag{8.3}
\]
\[
R_\emptyset = \frac{|\{S_i | S_i = \emptyset \land G_i = \text{NIL}\}|}{|\{G_i | G_i = \text{NIL}\}|} \tag{8.4}
\]

where \( Q \) is the number of queries in the dataset, \( G \) is the gold standard annotations for the dataset (\(|G| = Q\)), \( G_i \) is the gold standard for query \( i \) (KB ID or NIL), \( S \) is the system generated entity sets (\(|S| = Q\)), \( S_i \) is the system generated entity set for \( i^{th} \) query, \( S_{i,j} \) is the system generated entity at \( j^{th} \) rank of \( i^{th} \) query, \( P_\emptyset \) is the percentage of system generated NIL entity sets (sets that are either empty or singleton containing NIL) that are correct (which correspond to NIL queries), and \( R_\emptyset \) is the percentage of NIL queries for which the system generated NIL entity sets.

Cornolti et al. [?] have defined many evaluation measures for wikifier (A2W systems). The precision and recall measures specified there are in line with the \( P_\emptyset \) and \( R_\emptyset \) evaluation measures respectively. Further Hoffart et al. [?] define EE Precision and EE Recall which coincides with \( P_\emptyset \) and \( R_\emptyset \) respectively.

**Evaluation with KBs:** NEI can be evaluated using two KBs with one KB being a subset of the other. Entity Linking is performed with a smaller KB and the NIL entities identified are evaluated by their presence in the larger (superset) KB. This strategy is used in many EL systems for NEI [?] and NEI evaluation [? ? ?]. We use the evaluation setup specified by Graus et al. [?]. Given the TAC KB, we randomly sample entities to yield a smaller KB referred as KB \(_s\). Now KB \(_s\) simulates the available knowledge at the present point in time, whilst KB represents the future state. By measuring how many entities we are able to detect in our corpus that feature in KB, but not in KB \(_s\), we can measure NEI. KB \(_s\) is created by taking random samples of 20% to 100% the size of KB (measured in entities), in steps of 20% (as a sliding window). We repeat each sampling step five times to avoid bias.

### 8.2.3 Reproducing NEI Approaches

**Reproducibility:** Table 8.2 shows \( A_{\text{micro}} \), \( P_\emptyset \) and \( R_\emptyset \) for the two NEI approaches on the test set. Threshold value for NIL entity (\( \tau \)) for TAGME system, learned with an SVM regression classifier trained on the NIL queries of development dataset, was 0.22. This is in line with the TAGME description [?]. We evaluated Ploch system with both linear and RBF kernels. Ploch reports \( A_{\text{micro}} \) of 0.62 for KB queries (queries for entities present in KB). While we got the same result, we found that moving from linear kernel to RBF kernel, \( A_{\text{micro}} \) dropped from 0.62 to 0.34 for KB queries. On NIL queries, Ploch reports \( A_{\text{micro}} \) of 0.88 with RBF kernel. In our implementation we found this to be 0.52 with RBF kernel. We tried with the relaxed evaluation conditions as suggested in [? ], which took us near to the 0.88 result. But deviating from the evaluation measures defined in Section 8.2.2 will leave us with results that are not comparable. So we use 0.52 as \( A_{\text{micro}} \). On moving from RBF kernel to linear kernel, \( A_{\text{micro}} \) dropped from 0.52 to 0.44. This observation is reported in the paper [? ] too. Thus both the approaches are reproducible, though all results could not be reproduced.
Table 8.2: NEI Performance: Ploch system performance on NIL entities is better than that in-KB entities and also that of TAGME system. Use of dedicated NIL entity classifiers which uses NEI word features improves NEI performance of standard EL systems.

<table>
<thead>
<tr>
<th>NEIApproach</th>
<th>( A_{\text{micro}} )</th>
<th>( P_{\emptyset} )</th>
<th>( R_{\emptyset} )</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ploch(_{\text{KBlin}})</td>
<td>0.62</td>
<td>0.63</td>
<td>0.89</td>
<td>0.74</td>
</tr>
<tr>
<td>Ploch(_{\text{NILlin}})</td>
<td>0.44</td>
<td>0.70</td>
<td>0.81</td>
<td>0.75</td>
</tr>
<tr>
<td>Ploch(_{\text{KBbrf}})</td>
<td>0.34</td>
<td>0.63</td>
<td>0.49</td>
<td>0.55</td>
</tr>
<tr>
<td>Ploch(_{\text{NILbrf}})</td>
<td>0.52</td>
<td>0.60</td>
<td>0.94</td>
<td>0.73</td>
</tr>
<tr>
<td>TAGME(_{p})</td>
<td>0.47</td>
<td>0.62</td>
<td>0.80</td>
<td>0.70</td>
</tr>
<tr>
<td>TAGME(_{r})</td>
<td>0.45</td>
<td>0.62</td>
<td>0.73</td>
<td>0.67</td>
</tr>
</tbody>
</table>

Figure 8.3: NEI Accuracy by KB-subKB evaluation method: the Ploch system does slightly better than the TAGME system in NEI performance by this evaluation.

**Performance:** As this is the first reporting of \( A_{\text{micro}} \), \( P_{\emptyset} \) and \( R_{\emptyset} \) of TAGME system on this dataset, we do not have a benchmark. However we compare it to that of Ploch system performance. As TAGME system is designed to predict in-KB entities, we compare the in-KB results of Ploch system namely Ploch\(_{\text{KBlin}}\) and Ploch\(_{\text{KBbrf}}\) with TAGME results TAGME\(_{p}\). Here we see that \( P_{\emptyset} \) remains in the range of 0.63 ± 0.1. \( R_{\emptyset} \) remains higher in Ploch system compared to TAGME. With linear kernel Ploch system gave \( A_{\text{micro}} \) of 0.62 whereas TAGME system’s \( A_{\text{micro}} \) was 0.46 ± 0.1. Comparing the performance on NIL entities the F-measure (Eq. 8.2.2) of Ploch system fares better than that of TAGME system. Further the NIL-results are better than in-KB results in Ploch system, all proving that NEI features and separate NIL classifier help achieve better NEI.

**Alternate evaluation:** On evaluating with subset and master-set KBs, we report the accuracy of NIL predictions as Accuracy\(_{\emptyset}\). Accuracy\(_{\emptyset}\) is calculated by taking the set of correct predictions (true positives), and linking each mention to the referent entity in the super-set KB. This gives the fraction of newly discovered entities. Figure 8.3 shows the average Accuracy\(_{\emptyset}\) of TAGME and Ploch system, across the five samples on the y-axis with varying subKB sizes on the x-axis. Here we see that the Accuracy\(_{\emptyset}\) varies from 36% to 89% across 20% to 90% of KB size. Similar gradual improvement was observed.
in precision of the systems with increasing KB size. Recall of the systems increased with increasing sub-KB size, which could be attributed to a broader coverage in KB. This result is in line with that of Graus et al. [9]. We also observe that Ploch system does slightly better than TAGME system in NEI performance by this evaluation.

8.2.4 Observations from Re-implementations

In this section we look at possible improvements to the NEI approaches of EL systems based on the learnings from our analysis and re-implementations.

NEI approach types: Systems using supervised machine learning method for NEI were found to perform better than systems using thresholding and systems using heuristic methods. Heuristic approaches like empty candidate sets were found to perform poorly in high recall systems. Thresholding has been used for NEI starting from NEI approach of Bunescu and Pasca [9]. Hoffart et al. [9] note three shortcomings with thresholding systems. (i) Its empirical quality is not that good. (ii) Fixing one global value for \( \tau \) may be difficult and may affect NEI decisions on local mentions. (iii) Fixing different \( \tau \) values for different kinds of contents may need frequent re-tuning on appropriate training data. Basically, thresholding systems make a trade-off between precision of linking the in-KB entities and NEI performance, in choosing the \( \tau \) value. For example, TAGME system assigns NIL to a mention when \( \rho < \tau \). In our re-implementation, we find that setting a higher \( \tau \) results in more mentions predicted as NIL entity. Higher values of \( \rho \) (almost nearing \( \tau \)) was found to be a good indicator of new entity. Similar result was observed by Graus et al. [9] also, who showed that higher \( \rho \) value is an effective signal to separate noise from entities that are worth including in a KB. Thus we can conclude that, on EL systems using thresholding which are trained for optimum performance, mentions predicted as NIL entity with high value of \( \rho \) (almost nearing \( \tau \)) are good candidates for improving NEI performance.

NEI Features: Word features or lexical similarity of mention and context [9] to candidate entities, is the most popular feature used in NEI approaches. We observe that word features lead to better recall while meta-data features like prior probability, link probability and coherence lead to better precision. In our re-implementation, (Table 8.1, row ‘KB features’) we find that Ploch system uses all word features, while TAGME system uses only three of them. In ‘Disambiguation features’, Ploch system uses more word features compared to TAGME system which uses more meta data features. In Table 8.2, \( R_\theta \) is higher for Ploch\textsubscript{NIL} (both linear and RBF) than TAGME\_\textsubscript{\tau}. The increased recall could be due to the higher word overlap with KB. We find the system with better recall does better NEI. This result was observed by Graus et al. [9] also. Thus we can conclude that EL system’s NEI performance can be improved by dedicated classifier (or clusterer) using word features.

Evaluating NEI: NEI evaluation is a challenging task. Hoffart et al. [9] create a labeled dataset for the task with manual cleaning. Graus et al. [9] also report manual evaluation while Hachey et al. [9] use crowd sourcing. In this regard, NEI evaluation with two KBs, one KB being a subset of the other, is promising [9] and is especially well suited for the unsupervised NEI approaches [9].
Reproducing published results: We had to adapt the disambiguation function of TAGME and define the relatedness between two pages \( p_a \) and \( p_b \) as shown in Eq. (8.5)

\[
rel(p_a, p_b) = 1 - \delta(p_a, p_b)
\]  

(8.5)

This was needed as, when \( p_a \) and \( p_b \) are identical pages (\( \delta(p_a, p_b) = 0 \)) relatedness score \( rel(p_a, p_b) \) becomes 1. Otherwise \( rel(p_a, p_b) \) is a score between 0 and 1. Though TAGME \([?]\) and Milne & Witten \([?]\) systems use \( \delta \) to measure relatedness between two pages, we used \( 1 - \delta \) in our system for the above reason. Later in a personal communication, TAGME team confirmed that this adaptation was correct and required.

8.3 Conclusion

New Entity Identification is the process of identifying entities to be added to a Knowledge base. In this study we analyzed 31 NEI approaches and re-implemented two state-of-the-art approaches. Our attempts to reproduce published research results have helped us to get a deeper understanding of NEI task and get insights into recreating state-of-the-art results. We have presented a systematic investigation of the NEI task, by re-implementing representative systems from both the approaches in the literature. We have presented the first direct comparison of these approaches, analyzed the results and evaluated them using both evaluation measures method and KB-subKB method. From our experiments we find that NEI performance can be improved by using a dedicated classifier (or clusterer) with word features. Crowd sourcing is a promising way for new entity evaluation.

Although there are many research efforts in NEI, we believe that there are still many opportunities for substantial improvements in this field, for instance in identifying new entities from unstructured documents. Use of deep learning techniques have shown improvements in EL performance. Research to use these improvements to enhance NEI performance is an interesting direction. Further the increasing demand for constructing and populating domain-specific knowledge bases (e.g., in the domains of bio-medicine, entertainment, products, finance, tourism) makes domain-specific NEI important as well. We hope that the findings from this study will provide NEI researchers with a quick start in their efforts.
Chapter 9

Conclusion

This thesis aims to extract structured information on named entities from unstructured texts. Structured information about entities is stored in a knowledge base. Analyzing a given piece of text in English and predicting the words and phrases in it that potentially talk about a named entity is done by a Named Entity Recognizers (NER). Interpreting the potential named entities thus identified with knowledge-base entities and therefore ascertaining that they are named entities is Entity Linking. In this thesis we defined, designed and presented improved approaches to Entity Linking.

We saw the sub tasks of EL namely mention detection and mention disambiguation and how they depend on knowledge base for its successful completion in Chapters 4 and 5. We also saw in Chapter 6 how enhanced KB achieves higher precision in EL and how EL can contribute to KB enhancement in Chapter 8. Chapter 7 showed how by applying information linked entity improved document categorization task. We summarize these inter dependencies in Figure 9.1.

9.1 The Journey So Far

Mention detection can be improved by understanding the salience of the mention and semantic relation of the mention with respect to other entities of the document (Chapter 5). For mention disambiguation, EL systems primarily exploit two types of information: (1) degree of similarity between the
mention and the candidate entity string, and (2) coherence measure between the candidate entity and other entities mentioned in the vicinity of the mention in text. Given \( M \) is set of mentions in a document \( D \) such that \( M = \{m_1, ..., m_n\} \), \( G \) is a knowledge graph with nodes as entities \( E \) and edges as \( F \), the problem of mention disambiguation is to find an assignment \( \Lambda : M \rightarrow E \). This can be formulated as finding an assignment \( \Lambda^* \) maximizing the objective function:

\[
\Lambda^* = \arg \max_{\Lambda} \sum_{i=1}^{n} (\phi(m_i, e_{\Lambda_i}; G)) + \lambda \sum_{j \neq i, j=1}^{n} \psi(e_{\Lambda_i}, e_{\Lambda_j}; G))
\]

(9.1)

in which \( \phi(m_i, e_{\Lambda_i}; G) \) measures the contextual compatibility of mention \( m_i \) and entity \( e_{\Lambda_i} \). and \( \psi(e_{\Lambda_i}, e_{\Lambda_j}; G) \) measures the global coherence of an assignment \( \Lambda \).

In this thesis we have presented six approaches to EL problem based on varying conditions of text (Chapter \ref{5}) and varying conditions of KB (Chapters \ref{6}, \ref{8}). Text varied from well structured text with ample context like document (Section \ref{5.2}) to semi structured text with lesser context like tweets (Section \ref{5.4}) and to unstructured text almost no context as in search queries (Section \ref{5.6}). Challenges in KB involved NE not represented in the KB (Chapter \ref{6}) and NE represented with few facts in KB, i.e number of edges of the NE in KG is low (Chapter \ref{8}). In other words, approaches on varying conditions of text experimented with \( \phi(m_i, e_{\Lambda_i}; G) \) of the objective function and approaches on varying conditions of KB experimented with objective functions’ \( \psi(e_{\Lambda_i}, e_{\Lambda_j}; G) \).

The approaches presented in the thesis varied in supervision, Knowledge Graph, monolith versus modular (multi-module) and features. Figure \ref{9.2} represents the initial models which is a supervised approach to learn hand-crafted features for linking mentions to entities in Wikipedia Knowledge Graph \( G_1 \). The model consists of four sequential modules (Figure \ref{5.1}). Later models including ELDEN saw supervision confined to learning \( \phi(m_i, e_{\Lambda_i}; G) \) of the objective function while objective functions’s \( \psi(e_{\Lambda_i}, e_{\Lambda_j}; G) \) is learned in an unsupervised way. Figure \ref{9.3} represents these final models which uses a mixture of hand-crafted and machine learned features for linking mentions to entities. The Knowledge Graph \( G_2 \) is expanded with entities over and above those in Wikipedia, primarily using words and phrases frequently occurring in Wikipedia corpus. The model can be easily extended to a knowledge graph \( G_3 \) containing entities from any corpus as is demonstrated using web-corpus in the thesis.
In Table 9.1 we compare the two models we presented in Figures 9.2 and 9.3.

<table>
<thead>
<tr>
<th>Supervision</th>
<th>Knowledge Graph</th>
<th>Model</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>$EL_{initial}$</td>
<td>Supervised</td>
<td>$G_1$</td>
<td>Multi module</td>
</tr>
<tr>
<td>$EL_{present}$</td>
<td>Supervised and unsupervised</td>
<td>$G_2$ such that $G_2 \supseteq G_1$</td>
<td>Multi module, $\psi$ learned unsupervised</td>
</tr>
<tr>
<td>$EL_{future}$</td>
<td>Unsupervised</td>
<td>$G_n$ such that $G_n \supseteq G_2$</td>
<td>Monolith</td>
</tr>
</tbody>
</table>

Table 9.1: Entity Linking approaches

9.2 The Journey Beyond

In Table 9.1 we compare the models on type of supervision, the Knowledge Graph they link to. Knowledge Graph of Wikipedia entities is designated as $G_1$, this graph extended with entities with Wikipedia corpus is $G_2$. While conventional EL systems were built from individual modules in a modular way, deep learning methods build a monolith. While conventional EL systems used hand crafted features, we envisage newer systems to learn the features without supervision. Incorporating these views, we envisage the next EL model as $EL_n$ presented in the last row of Table 9.1.

To summarize, this thesis studied the issues and challenges in named entity extraction from unstructured text and proposes a approach to solve it. It defines the problem boundaries, based on the challenges encountered. Based on the problems, the thesis acquaints the reader with the domain and presents related work in the concerned area. Next, it presents approaches to solve them, exploring ways to improve the knowledge base that powers it. We believe the improved methods of EL and KB enhancement suggested in this thesis can be leveraged by a large number of applications thereby enhancing the Semantic Web.
Appendix A

A.1 Annotation Guidelines

The task here is to find the salient named entities from the tweet.

By named entities, we mean all the phrases in the tweet that are uniquely identifiable with a name. For example, the name Dravid is a named entity of category Person. We are interested in only four categories of named entities, Person, Place, Organization and Event. Some examples for each category are given below:

1. Person - Dravid, Brendon McCullum, Nolan
2. Place - Delhi, India, MCG
3. Organization - Google, Apple, Indian Cricket Board, Blackcaps (New Zealand team)
4. Event - World cup 2015, Sa vs Nz match

In the annotation interface, enter your name and collection (either training set or inter-annotator set). You can see a tweet followed by a image. The training set consists of 10938 tweets. In the inter-annotator set, there are 60 tweets, with 20 each from 3 topics, namely Movie (tweets posted for 62nd national film awards in India, whose results were announced on March 24, 2015), Cricket (tweets from Cricket World Cup 2015) and Apple (tweets from launch of Apple iWatch on April 24, 2015). Below the image, you get to see a list of named entities to select. These named entities may not be precise or clean as they are identified by programs. For example, the entry @amlahash denotes the cricketer, Hashim Amla but is embedded with other characters. Still it can be considered as a valid named entity. Sometimes, you get to see more than one named entities falsely marked as one single entity (for instance, AppleAAPLMacBookAppleWatchAppleWatchEvent). In this case, you can mark them as Not Annotatable.

Your task now is to select the salient named entities and mark those named entities if you can see it in the image. Following table list some tips to do that for each category of named entities.

1. Person (Sangakkara) - Mark it if you can find the person Sangakkara in the image. Even a piece of text Sangakkara would suffice.
2. Place (MCG) - Mark it if you are confident that the place is Melbourne cricket ground. You can always google for similar images.

3. Organization (Blackcaps) - If you get to see any New Zealander in the picture or logo of the New Zealand, you can mark it.

4. Event (World cup 2015) - If you get to see any clue for World cup 2015, you can mark it.

   Every-time you select a named entity, you get to see another list of options that corresponds to it’s Wikipedia entry. Pick a entry which perfectly matches the Wikipedia page. For example, if the named entity is “CWC15”, it’s corresponding Wikipedia page is http://en.wikipedia.org/wiki/2015_Cricket_World_Cup and the KB entry you choose is whatever that comes after http://en.wikipedia.org/wiki/ which is 2015_Cricket_World_Cup. Since this list is also computed by a program, there could be a situation where none of the entries makes any sense to the named entity. If such is the case, you can mark it as ‘None’.

   If you see a sarcastic tweet, do all the above and put ‘S’ in the comment sections. For example a morphed image of the NE or tweet-text contradicting the NE (as in calling victory a defeat) should be marked S. If you see an advertisement or pointless tweet (tweet not related to cricket (for training dataset) or any of the three topic (for inter-annotator set)), you need not annotate named entities and mark P in the comment section. If a tweet appears like a retweet (repeating tweet text and/or tweet image), you don’t need to annotate, mark D in the comment section.

   All the best!
Appendix B

B.1 WLM Coherence Measure

Given a graph \( G = (E, F) \), the WLM coherence measure \( \psi_{wlm}(e_i, e_j) \) between two entities \( e_i \) and \( e_j \) is defined as follows.

\[
\psi_{wlm}(e_i, e_j) = 1 - \frac{\log(\max(|C_{e_i}|, |C_{e_j}|)) - \log(|C_{e_i} \cap C_{e_j}|))}{\log(|E|) - \log(\min(|C_{e_i}|, |C_{e_j}|))}
\]

where \( C_e \) is the set of entities with edge to entity \( e \). Intuitively, WLM assumes that entities with common edges are related. For tail entities with less number of edges, lesser is the chance of common edges, leading to poor performance of WLM.

B.2 Entity distribution by Degree in KG

In KGs, all entities are not equally well connected. Figure B.1 shows degree (edge count) vs. number of entities having that degree in Wikipedia KG. We see that few entities have many edges and many entities have few edges. For example, new articles for named entities in Wikipedia are first created as stub pages. These entities are less popular and are examples of tail entities. They are not as well connected to other entities in KB as head (or popular) entities. These pages lack the desired encyclopedic coverage and connections to related entities. As the entity’s prominence increases in public domain, the article also gets richer in content and connections. Entities that have many edges are called head entities. Head entities are well connected to other entities in KB.

The number of tail entities in a KB is many folds larger than the number of head entities. In Figure B.1, we find that entities with 500 edges or less make up more than 90% of Wikipedia.
Figure B.1: Plot of edge-count versus number of entities in Wikipedia with that edge-count. As edge count increases, number of entities decreases drastically, leading to few entities with many edges (head entities) and many entities with few edges (tail entities), showing the long-tail effect.
List of publications

(listed in reverse chronological order)

- Journals


- Conferences


  2. Shashank Gupta, Priya Radhakrishnan, Manish Gupta, Vasudeva Varma *Enhancing Categorization of Computer Science Research Papers using Knowledge Bases* In proceedings of the First Workshop on Knowledge Graphs and Semantics for Text Retrieval and Analysis (KG4IR), held in conjunction with ACM SIGIR 2017, August 11, 2017, Tokyo, Japan,


8. Priya Radhakrishnan, Vasudeva Varma *Extracting Semantic Knowledge from Wikipedia Category Names* in proceedings of the 3rd Workshop on Knowledge Extraction (AKBC 2013) at CIKM 2013, San Francisco, CA, USA.