IJCLA VOL. 1, NO. 1-2, JAN-DEC 2010, PP. 121-136
RECEIVED 30/11/09 ACCEPTED 16/01/10 FINAL 09/03/10

Linking Named Entities to a Structured Knowledge Base

KRANTHI REDDY. B, KARUNA KUMAR, SAI KRISHNA, PRASAD PINGALI, VASUDEVA VARMA

International Institute of Information Technology, India

ABSTRACT

The task of entity linking aims at associating named entities with their corresponding entries in a knowledge base. The task is challenging because entities, can not only occur in various forms, viz: acronyms, nick names, spelling variations etc but can also occur in various contexts. To extract the various forms of an entity, we used the largest encyclopedia on web, Wikipedia. In this paper, we model entity linking as an Information Retrieval problem. Our experiments using TAC 2009 knowledge base population data set show that an Information Retrieval based approach fares slightly better than Naive Bayes and Maximum Entropy.

1 INTRODUCTION

The rise of web 2.0 technology has provided a platform for user generated content through web blogs, forums etc. This has lead to information overload on the web and it has become an extremely difficult task for users to find the precise information they are looking for. Semi-structured knowledge bases [1] like Wikipedia [2] act as a rich source of information for

[1] Knowledge Base is a structured data base containing entries describing named entities.

[2] Wikipedia is a huge collection of articles. Each article is identified by a unique title. These articles define and describe events and entities.

various user needs and are important for a wide range of applications like search, named entity extraction, text mining etc. But the problem with such structured knowledge bases is that they have to be created manually and updated frequently. For example, " *Information like movies starring Tom cruise have to be updated frequently*". There is also the problem of inconsistency, erroneous values being fed and the information not being up to date.

Automatically updating structured knowledge bases from news articles is a possible solution to this problem since news articles contain the latest information. In view of this strategy, a need arises to address the task of linking named entities from news articles to entries in a knowledge base.

The problem of entity linking shares similarities with cross document co-reference resolution. The task of co-reference resolution is to determine whether two occurrences in a document correspond to the same entity or not. This task becomes more complex when we try to determine whether the instances of two entities across different documents co-refer or not. This is termed as cross document co-reference resolution[1]. This is a challenging problem because the documents can come from different sources and they might also have different conventions and styles[1].

Thus, co-reference resolution of entities across documents plays a critical role towards successfully updating knowledge bases. In 1996 Tipster III program identified cross document co-reference resolution as an advanced area of research since it could be used for multi document summarization and information fusion. It was also identified as one of the potential tasks for the sixth Message Understanding Conference (MUC-6) but was not included as a formal task since it was considered too ambitious at the time[8]. Bagga and Baldwin [1]presented a successful cross document co-reference resolution algorithm to resolve ambiguities between people bearing same names using vector space model.

Following this work, many more contributions have been made to the state of the art. Bhattacharya[2] and Hal Daume[10] construct generative models on how and when entities are shared between documents. Haghighi and Klein[9] propose a fully generative nonparametric Bayesian model which captures co-reference within and across documents. Mann[12] and many of the other previous approaches such as Gooi[7], Malin[11], Chen[3] employ unsupervised learning techniques. Malin[11] considers named-entity disambiguation as a graph problem and constructs a social network graph to learn the similarity matrix. Finin et al.[5] explore the

usage of Wikipedia, DBpedia and free base as a resource for cross document co-reference resolution.

The task of entity linking differs from cross document co-reference resolution in the following aspects. In cross document co-reference resolution, we have a set of documents all of which mention the same entity name. The difficulty lies in clustering these documents into sets which mention the same entity. Whereas, in entity linking, the same entity name could be referred to in different contexts and also using various forms like acronyms, nick names etc. Our problem is to link this named entity to an entry in the knowledge base if present.

Key contributions of our paper are

- We show how variations of an entity, extracted from Wikipedia can be used for linking named entities from news articles to entries in a knowledge base.
- We show that an Information Retrieval based approach is able to perform slightly better than Naive Bayes and Maximum Entropy.

In section 2 we describe the data set and evaluation metrics. We sketch our algorithm in section 3 and describe the experiments conducted in section 4. In section 5 we provide an analysis of our results. We conclude our work in section 6 with a description of our plan for future work.

2 Evaluation Data and Metrics

In this section we describe the data collection and the entity linking tasks and the evaluation metric used. Our experiments were conducted on the Knowledge Base Population(KBP) [3] data set provided as part of the KBP track at Text Analysis Conference [4](TAC) 2009. The KBP data set consists of a reference Knowledge Base (KB) and a document collection. The KB comprises of 818,741 entries where each entry (entity/node) can belong to a Person, Organization, Geo Political Entity or an unknown class. The document collection contains instances of, and information about the target entities for the KBP evaluation queries. A sample KB entry is shown in Fig.1.

KB entries include a name string, an entity type, a unique KB node id, a set of facts and disambiguation text describing the entity.

[3] http://apl.jhu.edu/∼paulmac/kbp.html
[4] http://www.nist.gov/tac/

```
<entity wiki_title="Bud_Abbott" type="PER" id="E0064214" name="Bud Abbott">
<facts class="Infobox actor">
<fact name="name">Bud Abbott</fact>
<fact name="birthname">William Alexander Abbott</fact>...
</facts>
<wiki_text>Bud Abbott William Alexander "Bud" Abbott (October 2, 1895 – April 24, 1974) was an American actor, producer and
comedian born in Asbury Park, New Jersey...
</wiki_text>
</entity>
```

Fig. 1. Knowledge Base Entry

The document collection consists of 1,289,649 data files that contain instances of, and information about the target entities for KBP evaluation queries. The source documents in this collection were taken from various news transcripts and news articles. A sample data file is shown in Fig.2.

```
<DOC>
<DOCID> APW_ENG_20071010.1447.LDC2009T13 </DOCID>
<DOCTYPE SOURCE="newswire"> NEWS STORY </DOCTYPE>
<DATETIME> 2007-10-10 </DATETIME>
<BODY>
<HEADLINE> Peter Fonda auctions memorabilia from `Easy Rider ' film; flag brings about $90,000 </HEADLINE>
<TEXT>
<P> Aside from items offered by the 67-year-old Fonda, the auction included memorabilia related to Peter Frampton, Elvis Presley and
Abbott and Costello.</P>
</TEXT>
</BODY>
</DOC>
```

Fig. 2. Document Collection Data file

The data file consists of a unique document id, the source from where the article has been taken, its headline, and a disambiguation text describing an entity or an event. This text is split into different paragraphs.

The task of entity linking is to determine for each query, if a KB entry exists in the knowledge base. And if it does which KB entry it refers to. A query will consist of a name-string and an associated document-id from the document collection providing context for the name-string. For convenience we refer to name-string as "query string". Query strings can occur as multiple queries using different name variants or in multiple documents. Each query must be processed independently. A sample query is shown in Fig.3.

```
<query id="EL24"><name>Abbott</name><docid>AFP_ENG_19960413.0028.LDC2007T07</docid></query>
<query id="EL25"><name>Abbott</name><docid>APW_ENG_20080725.0086.LDC2009T13</docid></query>
<query id="EL26"><name>Abbott</name><docid>AFP_ENG_20030724.0396.LDC2007T07</docid></query>
```

Fig. 3. Sample Queries

Since the documents can come from different sources, various name variations like acronyms and nick names etc could refer to the same query string. They might also occur in different contexts which makes the problem a challenging one.

Table 1 shows that there are 15 queries with “Abbott/Abbot” as the query string, but they refer to different KB entries which belong to different classes. The query string is associated with 15 different data files showing how varied the context is.

Table 1. Sample Queries

Query string	**KB-id**	**KB Entry title**	**No. of Queries**	**No. of diff. data files**	**Class**
Abbot	E0064214	Bud Abbott	1	1	Person
Abbott	E0064214	Bud Abbott	4	4	Person
Abbott	E0272065	Abbott Lab.	9	9	Unknown
Abbott	E0003813	Abbot, Texas	1	1	Geo-political entity

The following two examples show how varied the context can be.

Context 1: *A spokeswoman for Abbott said it does not expect the guidelines to affect approval of its Xience stent, which is expected in the second quarter.*

Context 2: *Aside from items offered by the 67-year-old Fonda, the auction included memorabilia related to Peter Frampton, Elvis Presley and Abbott and Costello.*

In context 1 “Abbott” refers to “Abbott Laboratories” whereas in context 2 it refers to “Bud Abbott”. The above example shows that the task of entity linking is a challenging one.

To evaluate the effectiveness of the system, we use Micro-Average Score (MAS). MAS is the official metric for evaluating systems performance at TAC 2009. The micro-average score is the precision over all the queries. It is calculated using the following equation.

$$Micro\ Average\ Score = \frac{No.of\ correct\ responses}{No.of\ Queries} \quad (1)$$

Similarly micro-average score can be calculated for nil valued queries and Non-nil valued queries. From Table 2, system output is correct 3 out of 6 times. Hence the Micro Average Score is 3/6 = 0.5 .

Another metric that can be used to evaluate the entity linking task is the Macro-Average Score. In this metric, precision is calculated for each entity (nil and non-nil) and an average is taken across the entities. The main problem with such a metric is that it might be biased towards the system's output. It would be unstable with respect to low-mention-count query entities. The example below explains the calculation of Macro-Average Score.

Table 2. System output for a set of query strings

Query string	**KB-id**	**system output**
Abbott	1	1
Abbott	1	101
Abbott	1	1
Abbott Labs	2	101
Abbott Laboratories	2	nil
Abbott Labs	2	2

From Table 2, the entity corresponding to the KB node with ID=1 was linked correctly 2 of 3 times for a precision of 0.67. The entity with ID=2 was linked correctly 1 of 3 times for a precision of 0.33.The macro-averaged precision is 0.5. $\{(0.67+0.33)/2\}$

3 Our Approach

We break the entity linking task into 3 sub tasks.

1. **Preprocessing:** Since the queries for entity linking can have different name variations, we need to have a knowledge repository of all

the various forms possible for an entity. During this step we build a knowledge repository that contains vast amount of world knowledge for these entities. To create this we can use the web. But parsing and extracting knowledge from the web is a tedious task because of its sparseness and unstructured format. Hence we turned our attention to Wikipedia, which is one of the largest semi-structured knowledge bases on the web[17].

The advantages of using Wikipedia compared to the web or any other resource is that

- It has better coverage of named entities [18]. And since our knowledge base presently has only named entities, Wikipedia acts as a perfect platform for creating our knowledge repository.
- Redirect pages can be used to induce synonyms [18].
- Disambiguation pages can be used to generate a list of candidate targets for homonym resolution[14].
- On analyzing random pages from Wikipedia, we found that the bold text from the first paragraph is a variation of the title. These variations in general are full names, alias names and nick names of the title.
- With over 3 million articles Wikipedia is appropriately sized and big enough to provide us sufficient information to create our knowledge repository.

A lot of previous work on wikipedia mining[6, 15, 16] confirms the fact that valuable information can be mined from Wikipedia .

We use redirect pages[5], disambiguation pages [6] and bold text from the first paragraph to create our knowledge repository, which is simply a collection of different variations of an entity. These heuristics help us in identifying synonyms, homonyms, abbreviations, frequent misspellings and alternative spellings of an entity. Even though we are handling different valid variations, each of the above variations can be misspelled as well. In order to identify these spelling variations we generate metaphonic codes for all the variations using metaphone algorithm[4].

2. **Candidate List Generation:** The entity linking task first determines whether a KB entry exists for a given query string. The query string

[5] A redirect page in Wikipedia is an aid to navigation, it contains no content but a link to another article (target page) and strongly relates to the concept of target page.

[6] Disambiguation pages are specifically created for ambiguous entities, and consist of links to articles defining the different meanings of the entity.

is then searched on the titles of KB nodes and Wikipedia articles in the following two ways.

(a) **Phrase Search:** In this method we see if the exact phrase of the query string or the expanded form of the acronym is present in the article title or not. We add node-ids to candidate list only if we find the exact phrase in article titles. In another variation of phrase search, we allow for a difference of one between the number of tokens in article title and query string. We term this extra token as noise.
For example, If the given query is "UT" and we find the expanded form from our knowledge repository to be "University of Texas"; in exact phrase search we would be retrieving node-ids that have exact phrase "University of Texas" present in the title. Whereas in phrase search with noise we would be retrieving nodes that have the titles "University of Texas at Austin, University of Texas at Dallas" as well.

(b) **Token search:** In this method we do a boolean "AND" search of all the tokens of the query string or the expanded form of the acronym in the article title. If all the tokens are present in an article title we add those node-ids to the candidate list. Another variation is to search with noise.
The difference between phrase search and token search is that in phrase search the token order is constrained where as in token search just the presence of each token is vital and not the order.
For example, If the given query is "CCP" and we find the expanded form from our knowledge repository to be the "Chinese Communist Party"; in token search we would be retrieving nodes with the titles either as "Chinese Communist party" or "Communist Party of China". Note that the entry with the title "Communist Party of China" will not be found as a candidate item in phrase search.

All the KB nodes and Wikipedia articles which satisfy one of the above conditions are added to the candidate list. The addition of Wikipedia articles to the candidate list helps us in the identification of nil valued queries. That is, for a given query if our algorithm maps to the Wikipedia article from the candidate list, we can confirm the non-presence of an entry in KB describing the query string.
The query strings can be categorised as either acronyms or any of the other variations like alias names, nick names etc. If all the characters present in the query are uppercase we consider it to be an acronym

. We follow different approaches for processing the two categories. The algorithm for handling these two cases is as follows.

(a) **Not an Acronym:** If the given query is not an acronym we search for the query string terms directly in the title of the KB entries. If a hit is found we add that entry's node-id to the candidate list. However, if no hits are found, we look for variations of the query string in the knowledge repository and then use them to search the KB and Wikipedia titles. If any hits are found we also add those node-ids to the candidate list.

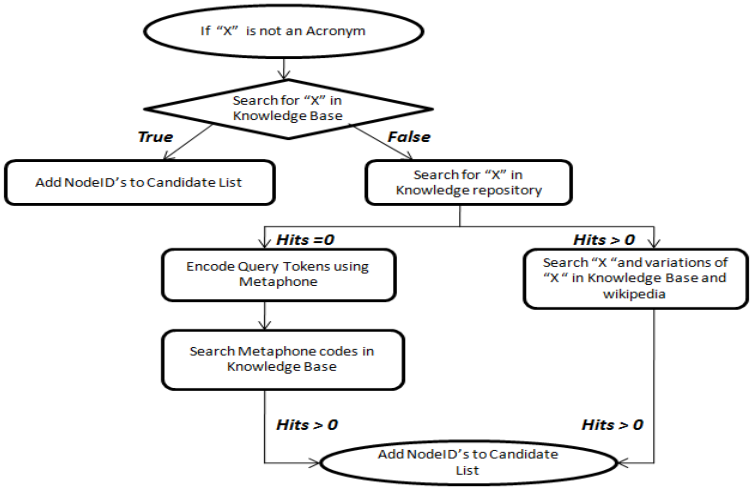

Fig. 4. Flowchart when query is not an Acronym

If no variations are found in the knowledge repository as well, we assume that the query string might have been written using a different spelling. We then generate the metaphone code for each token present in the query. Using these metaphone codes we again search the KB. The flowchart of algorithm is presented in Fig.4.

(b) **Acronym:** If the given query is an acronym we try to get the expanded form from the document content which has been given as disambiguation text for the query string. To find the expanded form, we remove stop words from this disambiguation text and

use an N-Gram based approach. In our N-Gram approach if the length of the acronym is "N" characters, we check if "N" continuous tokens in the disambiguation text have the same initials as the characters of the acronym. If we are successful in finding the expanded form from the disambiguation text, we search the KB using this expanded form. If any hits are found we add the entry's node-id to the candidate list. If we don't find the expanded form from the disambiguation text we search the knowledge repository for the acronym. If the expanded form is found in the knowledge repository, we search the KB and Wikipedia titles using this expanded form and the acronym. If any hits are found we add the entry's node-id to the candidate list. The flowchart of algorithm is presented in Fig.5.

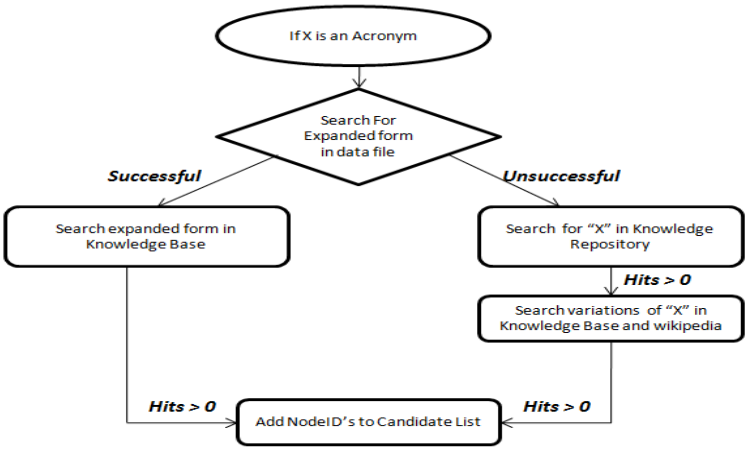

Fig. 5. Flowchart When query is an Acronym

Refining the Candidate list: Articles in KB and Wikipedia are uniquely identified by their titles. With KB being a subset of Wikipedia, for a particular entity the title of articles that describe them would be the same. Thus, if duplicate entries are found, priority is given to KB articles.

3. **Calculating Similarity Score:** We return nil if there are no items in our candidate list or when there are node-ids that belong only to

Wikipedia. If there is only one node-id belonging to KB, it could possibly mean we have only one entry describing the query string. We return this node-id as the possible map.

If there is more than one entry in our candidate list we find the best map using one of the following approaches.

Classification Approach: We have conducted our experiments using Naive Bayes[13] and Maximum Entropy algorithms present in Rainbow Text Classifier[7].

If we consider all the possible candidate items as different classes, we need to find which class is the best map for our query string. Hence, we view this problem as a classification problem with each candidate being a class. Therefore, we built classification models using Naive Bayes and Maximum Entropy algorithms with bag of words as a feature. We use the text describing the candidate item(entity) provided in the KB to train the classification models. We then give the disambiguation text provided along with the query string as test document. This test document is classified into one of the classes and the score obtained is the likelihood of the test document belonging to that class.

Information Retrieval Model: In Information Retrieval, the aim is to retrieve the documents that are closest match for a given natural language query. In our approach, we index each candidate item as a separate document using Lucene[8]. We then form a query from the disambiguation text. Query formulation plays an important role in the success of this approach. While generating the query we try and reduce unwanted tokens. We also try to boost the tokens that seem to be most relevant to our query string. Since the provided disambiguation text has been tagged clearly into different paragraphs, we consider only those paragraphs where the query string is present. The motivation behind this is to capture the context surrounding our query string. We form a boolean "OR" query of all the tokens generated from the disambiguation text neglecting the stop words . We also boost the tokens that are within a window size of 5 terms on either side of the query string. We do this because the tokens closer to the query string are the more prominent tokens describing our entity than the terms that are far off. This is shown in the Fig.6.

[7] http://www.cs.cmu.edu/∼mccallum/bow/rainbow/

[8] Lucene is a high-performance, full-featured text search engine. http://lucene.apache.org/java/docs/.

 text <Query boosting terms> "Query Terms" <Query boosting terms> text

Fig. 6. Token boosting during Query Formulation

If the result node-id belongs to KB, then we return it as the map for the query string. But if the node-id belongs to Wikipedia we return nil, because we don't have an entry in the KB describing our query string.

4 EXPERIMENTS

To generate the candidate list we use one of the following: exact phrase search, phrase search with noise, token search, or token search with noise. Once the candidate list is generated, we use different algorithms to calculate the similarity score between the disambiguation text of the query string and the disambiguation text of the candidate item entries. If more than one item is present in the candidate list belonging to KB and/or Wikipedia, we calculate the similarity score using Naive Bayes, Maximum Entropy from Rainbow Text Classifier or by using an Information Retrieval approach. The algorithm is evaluated using the metrics described in section 2. Table 3 contains the scores for each experiment conducted.

The Micro-Average Score obtained through our algorithm outperforms all the systems submitted at TAC 2009. The average-median score over all the 35 runs submitted at TAC 2009 is 71.08% and the base line score is 57.10% when nil is returned for all the queries. Our best algorithm outperforms median score by as much as 11% and the base line score by 25%.

5 ANALYSIS

Since we have followed a two step process to determine whether for a given query string an entry exists in the knowledge base or not. Therefore for each of these steps we analyze the number of queries that are being incorrectly mapped. For token search with noise, we found that we were unable to find a map for 6.81% (266 of 3904) queries during the candidate list generation, which means that our heuristics failed to capture some query string variations of nicknames, full names, acronyms etc.

Table 3. Results of Various Experiments. IR = Information Retrieval, NB = Naive Bayes, MaxEnt = Maximum Entropy

Alg.	**Noise**	**Phrase/Token search**	**Micro-Average Score**	**nil-valued precision**	**Non-nil valued precision**	**Macro-Average Score**
NB	**1**	**Word Search**	**81.43**	**85.42**	**76.12**	**75.38**
NB	1	Phrase Search	81.25	85.37	75.76	75.10
NB	0	Phrase Search	81.12	85.91	74.75	75.45
NB	0	Word Search	80.87	85.51	74.69	74.96
MaxEnt	**0**	**Word Search**	**78.82**	**87.66**	**67.04**	**75.61**
MaxEnt	1	Word Search	78.46	87.53	66.39	75.58
MaxEnt	0	Phrase Search	78.38	87.93	65.67	76.08
MaxEnt	1	Phrase Search	78.23	87.44	65.97	75.90
IR	**1**	**Word Search**	**82.25**	**86.32**	**76.84**	**75.70**
IR	1	Phrase Search	82.17	86.41	76.54	75.39
IR	0	Phrase Search	81.81	86.90	75.04	75.46
IR	0	Word Search	81.76	86.45	75.52	75.54

While 10.93% (427) were being wrongly mapped during similarity score calculation.

For non-nil valued queries Fig.7 plots the comparison of Precision vs Top "N" search results for the 3 algorithms. It can be seen clearly that as we consider a higher number of hits, the probability of finding the correct map for the query string in the hits list increases. It shows that Information Retrieval and Naive Bayes perform consistently much higher than Maximum Entropy.

Thus we can conclude that the combination of token search with noise for candidate list generation and the Information Retrieval approach for similarity score calculation give the best result. The reason for this approach outperforming all the others is that we are able to generate more candidate items. Though this might also generate more false negatives, but the removal of unwanted paragraphs as noise and the query boosting technique used while calculating similarity score negates this effect.

6 Conclusion and Future Work

In this paper, we explored Information Retrieval and classification based techniques for linking named entities from news articles to entries in a

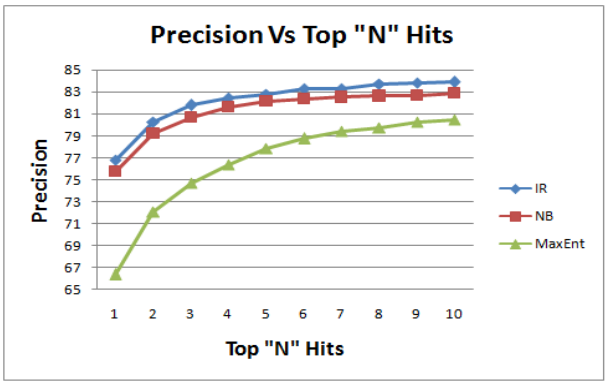

Fig. 7. Precision Vs Top “N” hits.
IR = Information Retrieval, NB = Naive Bayes, MaxEnt = Maximum Entropy

knowledge base. We showed how variations of an entity can be extracted from Wikipedia and used for entity linking. We showed that an Information Retrieval based approach is able to perform slightly better than Naive Bayes and Maximum Entropy approaches. We believe that our approach is promising because, Wikipedia is constantly growing and being updated frequently. With its continuous growth and contribution from users we are guaranteed high quality information. There can be many extensions to the current work. First, using Wikipedia in a better way to create our knowledge repository. We can make use of the infobox tables to extract name variations, nick names etc. Secondly, since news articles always contain the latest information about an entity, we can extract attribute value pairs from them and append them to our KB facts. This will be particularly useful when certain facts keep changing frequently. For example, the number of test matches played by Sachin Tendulkar, number of runs scored by him etc. Thirdly, we can make use of other online resources like DBpedia and Freebase to create our knowledge repository.

7 AKNOWLEDGEMENTS

The authors wish to thank TAC 2009 organisers for providing the data set without which this paper would have never come. The authors also wish to thank anonymous reviewers for their positive feedback. The authors in particular wish to thank Abhilash, Kiran, Praneeth and Arafat for their kind support, help and guidance.

REFERENCES

1. A. Bagga and B. Baldwin. Entity-based cross-document coreferencing using the vector space model. In *Proceedings of the 17th international conference on Computational linguistics-Volume 1*, pages 79–85. Association for Computational Linguistics Morristown, NJ, USA, 1998.
2. I. Bhattacharya and L. Getoor. A latent dirichlet model for unsupervised entity resolution. In *SIAM International Conference on Data Mining*, pages 47–58, 2006.
3. Y. Chen and J. Martin. Towards robust unsupervised personal name disambiguation. *Proceedings of EMNLP and CoNLL*, pages 190–198, 2007.
4. S. Deorowicz and M.G. Ciura. Correcting spelling errors by modelling their causes. *International journal of applied mathematics and computer science*, 15(2):275, 2005.
5. T. Finin, Z. Syed, J. Mayfield, P. McNamee, and C. Piatko. Book Title: Proceedings of the AAAI Spring Symposium on Learning by Reading and Learning to Read Date: March 23, 2009.
6. E. Gabrilovich and S. Markovitch. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 6–12, 2007.
7. C.H. Gooi and J. Allan. Cross-document coreference on a large scale corpus. In *Proceedings of HLT/NAACL*, 2004.
8. R. Grishman. Whither written language evaluation. In *Proceedings of the Human Language Technology Workshop*, pages 120–125, 1994.
9. A. Haghighi and D. Klein. Unsupervised coreference resolution in a nonparametric bayesian model. In *ANNUAL MEETING-ASSOCIATION FOR COMPUTATIONAL LINGUISTICS*, volume 45, page 848, 2007.
10. Hal Daume III and Daniel Marcu. A bayesian model for supervised clustering with the dirichlet process prior. *J. Mach. Learn. Res.*, 6:1551–1577, 2005.
11. B. Malin. Unsupervised name disambiguation via social network similarity. In *SIAM SDM Workshop on Link Analysis, Counterterrorism and Security*. Citeseer, 2005.
12. G.S. Mann and D. Yarowsky. Unsupervised personal name disambiguation. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 33–40. Association for Computational Linguistics Morristown, NJ, USA, 2003.
13. A.K. McCallum. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering, 1996.
14. R. Mihalcea. Using wikipedia for automatic word sense disambiguation. In *Proceedings of NAACL HLT*, volume 2007, 2007.
15. D. Milne, O. Medelyan, and IH Witten. Mining domain-specific thesauri from wikipedia: A case study. In *IEEE/WIC/ACM International Conference on Web Intelligence, 2006. WI 2006*, pages 442–448, 2006.

16. K. Nakayama, T. Hara, and S. Nishio. Wikipedia mining for an association web thesaurus construction. *Lecture Notes in Computer Science*, 4831:322, 2007.
17. M. Remy. Wikipedia: The free encyclopedia. *Reference Reviews*, 16.
18. T. Zesch, I. Gurevych, and M. Muhlha user. Analyzing and accessing Wikipedia as a lexical semantic resource. *Data Structures for Linguistic Resources and Applications*, pages 197–205, 2007.

KRANTHI REDDY. B
INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY,
HYDERABAD, INDIA
E-MAIL: <BKREDDY@RESEARCH.IIIT.AC.IN>

KARUNA KUMAR
INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY,
HYDERABAD, INDIA
E-MAIL: <KARUNA_KY@STUDENTS.IIIT.AC.IN>

SAI KRISHNA
INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY,
HYDERABAD, INDIA
E-MAIL: <SAIKRISHNA@RESEARCH.IIIT.AC.IN>

PRASAD PINGALI
INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY,
HYDERABAD, INDIA
E-MAIL: <PVVPR@IIIT.AC.IN>

VASUDEVA VARMA
INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY,
HYDERABAD, INDIA
E-MAIL: <VV@IIIT.AC.IN>