Improving Candidate Generation for Entity Linking

Yuhang Guo[1], Bing Qin[1,⋆], Yuqin Li[2], Ting Liu[1], and Sheng Li[1]

[1] School of Computer Science and Technology,
Harbin Institute of Technology, Harbin, China
[2] Beijing Information Science and Technology University, Beijing, China
{yhguo,bqin,tliu,sli}@ir.hit.edu.cn,
li.yuqin@trs.com.cn

Abstract. Entity linking is the task of linking names in free text to the referent entities in a knowledge base. Most recently proposed linking systems can be broken down into two steps: candidate generation and candidate ranking. The first step searches candidates from the knowledge base and the second step disambiguates them. Previous works have been focused on the recall of the generation because if the target entity is absent in the candidate set, no ranking method can return the correct result. Most of the recall-driven generation strategies will increase the number of the candidates. However, with large candidate sets, memory/time consuming systems are impractical for online applications. In this paper, we propose a novel candidate generation approach to generate high recall candidate set with small size. Experimental results on two KBP data sets show that the candidate generation recall achieves more than 93%. By leveraging our approach, the candidate number is reduced from hundreds to dozens, the system runtime is saved by 70.3% and 76.6% over the baseline and the highest micro-averaged accuracy in the evaluation is improved by 2.2% and 3.4%.

Keywords: Natural Language Processing, Information Extraction, Entity Linking, Candidate Generation, Candidate Pruning.

1 Introduction

Entity Linking (EL) is the task of identifying the target entity which a name refers to. It can help text analysis systems to understand the context of the name in-depth by leveraging known information of the entity. On the other hand, new knowledge about this entity can be populated by mining information from the context. Figure 1 illustrates entity linking can help question answering: knowing the name *Washington* refers to actor `Denzel Washington` (rather than `George Washington` or the `State of Washington`) in the question: *Who did Washington play in Training Day*, one can find the corresponding answer (*Detective Alonzo Harris*) directly in the knowledge base.

⋆ Corresponding author.

E. Métais et al. (Eds.): NLDB 2013, LNCS 7934, pp. 225–236, 2013.

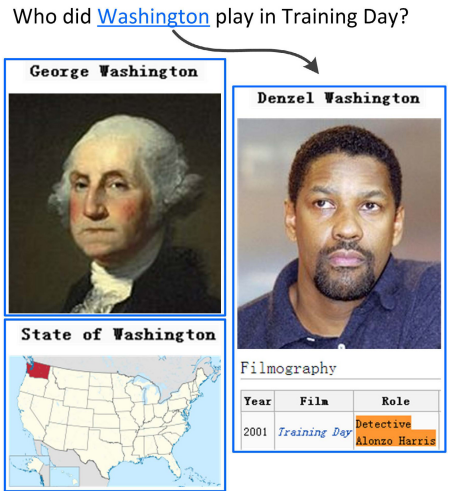


Fig. 1. An example of entity linking

EL can be broken down into two steps: candidate generation and candidate ranking. The first step generates a set of candidate entities of the target name and the second step ranks the candidates. Several ranking models have been proposed for the second step. However, few works have focused on the candidate generation step. Generating candidates is a critical step for the linking systems. If the target entity is not included in the candidate set, no ranking model can return the correct one.

A number of resources have been proposed to improve the generation recall [2,4,24,6]. By leveraging these resources, the number of the candidates can be very big. Take the target name *Washington* for example, the generation will return more than 600 candidates.

Bounding the number of the candidates is important in the applications of EL. Lessening the candidates will reduce time and memory costs of the ranking, and further make sophisticated time and memory consuming ranking models be practicable. How to generate small candidate sets under the premise of ensuring high recall is an interesting problem.

In this paper, we propose a novel candidate generation approach. In this approach, the generator first extracts the target name's co-reference names in the context. From this set the generator then selects the most reliable name (i.e. the least ambiguous name) to generate candidates by leveraging a Wikipedia-derived name-entity mapping. Next the generator prunes the candidates according to their frequencies and their similarity to the target name.

Experiment on benchmark data sets shows that our candidate generation can increase the recall and reduce the candidate number effectively. Further analysis shows that both the accuracy and the speed of the system can benefit from the proposed candidate generation approach, especially for the target names with large candidate set. The system runtime can be effectively saved over the baseline

candidate set. The highest accuracy in the evaluation is improved by 2.2% and 3.4%.

2 Related Work

EL is similar to Word Sense Disambiguation (WSD), a widely-studied natural language processing task. In WSD the sense of a word (e.g. bank: river bank or a financial institution) is identified according to the context of the word [10,20,15]. Both WSD and EL disambiguate polysemous words/names according to the context. The difference between the two tasks is in that, the disambiguation targets in WSD are lexical words whereas in EL are names. In WSD, the senses of words are defined in dictionaries, such as WordNet [18]. In EL, however, no open domain catalog has included all entities and all of their names. The study on WSD have a history of several decades[10,20,15]. Recently, as the development of the large scale open domain knowledge bases (such as Wikipedia, DBpedia[1,1] and Yago[23], etc.), EL has been attracting more and more attentions.

Early EL borrowed successful techniques in WSD: take each sense (candidate entity) as a class and resolve the problem by multi-class classifier[17,2]. However, in WSD a word usually has several senses but in EL a name may have dozens to hundreds of candidate entities. Under such high polysemy, the accuracy of the classifier cannot be guaranteed.

EL systems can be broken down into two steps: candidate generation and candidate ranking[11]. Early candidate generation approaches directly match the target name in the knowledge base[2]. Recently, several techniques have been proposed and have achieved certain success in recall.

- Substitute the target name to a longer name in the names co-reference chain in the context[4].
- If the target name is an acronym, substitute it with the full name in the context[4,24].
- Filter acronym expansions with a classifier[26].
- If the exact match fails, then use partial search[24] or fuzzy match[14] (e.g. return candidates with high Dice coefficient).

The candidate ranking is based on the similarity between the candidate entity and the context surround with the target name. A number of features have been proposed: Plain text[24]; Concepts, such as Wikipedia category[2], Wikipedia concept[9], topic model concept[12,22,26]; And neighboring entities, which include the entities mapped from unambiguous names[19] and the collectively disambiguated entities[4,13,8,22]. The entity-context similarity is measured in: cosine similarity[24], language model score[7] and the inner coherence among neighboring entities measured by link similarity[19,21,22] and collective topic model similarity[22]. Besides directly use these similarities for the ranking, machine learning methods has been applied to combine these similarities[19,27,5].

Sophisticated ranking models need heavy computation costs. For example, the time complexity of the list wise learning to rank method is exponential[3,25,27].

Collective disambiguation is NP hard[13]. Therefore, generating small candidate sets is important to these ranking models. However, little work to date has focused on the candidate pruning.

3 Candidate Generation Approach

An entity may be mentioned many times with different names in document. Some of the names are easier to be linked than others. For example, *Denzel Washington* is less ambiguous than *Washington*. In this paper we propose a context based candidate generation approach (CBCG). CBCG first detects co-reference names of the target name in the context. The co-reference names are the target entity's potential names, including acronym expansion, longer names and shorter names. Then the approach match probably the least ambiguous potential name in a Wikipedia-derived Name-Entity Mapping (NEM). Next the returned entities are filtered by their frequency and their similarity to the target name. To summarize, CBCG reduces the candidate number by leveraging three strategies: back-off, filter by frequency, and filter by similarity.

Using the back-off strategy, the most reliable name is first considered. The CBCG generator considers the next most reliable name only if the current name returns no candidate. Using the filter by frequency strategy, the generator set a volume threshold to the candidate set and low frequency candidate will be filtered. Using the filter by similarity strategy, the generator set a similarity threshold and the candidates with low similarity with the target name will be filtered. In the following of this section, we will describe the NEM construction, the potential name detection, and the candidate pruning in detail.

3.1 Name-Entity Mapping Construction

An entity may be mentioned in different name. Some of these name variations (or aliases, alternative names) represent the entity frequently, and some others not. Collecting as many name variations of the entity as possible involves the recall when this entity is referred to. Name-entity pairs and the co-occurrence frequency can be mined from the following Wikipedia structure:

- Page and redirect page title of the entity.
- Title of the disambiguation page which contains the entity.
- Anchor text which targets to the entity.
- Bold text in the first paragraph of the entity.
- Value of the name field (e.g. birth_name, nick_name, etc.) within Infobox,[1].

In this mapping, a name is mapped to all the entities it may refer to. For example, name *Washington* is mapped to `Denzel Washington`, `George Washington` and `State of Washington`, etc. All through this work, we use the Aug. 2, 2012

[1] A information structure of Wikipedia.

version of English Wikipedia dump, which contains more than 4.1 million articles[2]. In all, we extract 23,895,819 name-entity pairs with their co-occurrence frequencies. Summing up this frequency for the same entity, we can get the frequency of the entity in Wikipedia, which will be used in the following part of the linking system.

3.2 Potential Name Detection

We first apply forward maximum match algorithm on the context to extract all names that match in the NEM. Then we select the names which contains the target name (i.e. longer name) or is a substring of the target name (i.e. shorter name) as the potential name of the target entity.

Besides longer names and shorter names, potential name set also contains transformations of the target name. Because many all-capital names (e.g. *ARGENTINA*) cannot be matched in the NEM, we normalize the non-acronym all-capital words into Wiki-style[3]. We also substitute the state abbreviation names[4] (e.g. *CA*) in the document into the full forms (e.g. *California*).

Acronym target names (i.e. *ABC*) should be considered separately. The acronyms and their full names usually satisfy other constrains. For example,

- The full form is in front of the enclosed acronym (e.g. ... the newly formed All Basotho Convention (ABC))
- The acronym is in front of the enclosed full form (e.g. ... at a time when the CCP (Chinese Communist Party) claims ...)
- The acronym consists of the initial letters of the full name words (e.g. ... leaders of Merkel's Christian Democratic Union ... CDU ...)

These cases can be covered by several regular expressions.

Here we propose a novel acronym identification rule: a name string is an acronym if it satisfies all the following conditions:

- It contains no more than 4 letters.
- It contains no less than 2 upper case letters.
- It contains no more than 2 lower case letters.

According to the above rules, for example, *ABC* and *MoD* are identified as acronyms, and *Abbott* and *ARGENTINA* are not.

3.3 Candidate Pruning

The objective of this step is to minimize the set size of the candidates to be generated and maximize the possibility that the target entity is reserved in the set. The first strategy to reduce the number of the candidates is the back-off strategy: consult the most reliable potential name for the generation and

[2] `http://stats.wikimedia.org/EN/TablesWikipediaEN.htm`
[3] `http://en.wikipedia.org/wiki/MOS:TITLE#Composition_titles`
[4] `http://en.wikipedia.org/wiki/List_of_U.S._state_abbreviations`

consult the next most reliable potential name only if the current name returns no candidate.

Two points should be considered for the reliability of the potential name:

1. The number of the entities generated by this name. (N)
2. The probability of this potential name is a name of the target entity.(P)

According to our observation, longer name has a smaller N and higher frequency name has a higher P. In order to keep a small candidate set and a high recall at the same time, the considered potential name should be of both high frequency and long.

In this work, the potential names are first sorted by their types: longer names, normalized query names (including acronym expansion and Wiki-style normalization) and shorter names, and then by frequency in the same type.

The back-off strategy prunes candidates from name aspect. Whereas the following strategies prune candidates from the entity aspect. The filter by frequency strategy filter out the candidates with low frequency and the filter by similarity strategy filter out candidates with low similarity to the target name. We define the similarity between a name and an entity as follows: The target name n_t is similar to a candidate entity e if and only if at least one name (n_e) of this entity is similar to n_t.

Here we propose a novel name similarity measurement. The formula is

$$Sim(n_e, n_t) = \frac{\sum_{w \in n_e} Len(LCS(w, n_t))}{\sum_{w \in n_t} Len(w)} \tag{1}$$

where $Len(s)$ is the length of string s, $LCS(s_1, s_2)$ is the longest common string of s_1 and s_2. Note that this similarity is asymmetric.

Table 1. Notations

$\texttt{Sort}(\cdot)$	Returns a queue sorted by frequency.
$\texttt{Pop}(\cdot)$	Pop the top element in a queue.
$\texttt{Sim}(s_1, s_2)$	Return the similarity between string s_1, s_2.
$\texttt{E}(n)$	Returns the entities whose name matches n.
$\texttt{N}(e)$	Returns the names whose entity matches e.
n_t	Target name.
e_t	Target entity
N_p	Potential name set of e_t
C	Candidate entity set
Q_N	Potential name queue, sorted by frequency.
Q_E	Entity queue, sorted by frequency.

The candidate pruning strategies are combined in Algorithm 1. The symbols are described in Table 1 The candidate number is controlled by two parameters: a

similarity threshold is used to filter out the un-similar entities, and the candidate set volume threshold limits the maximum size of the candidate set[5].

```
input : target name n_t, potential name queue Q_N, similarity threshold p, and
        candidate set volume threshold T
output: candidate set C
C ← φ;
while C = φ and Q_N ≠ φ do
    n ← Pop(Q_N);
    Q_E ← Sort(E(n));
    i ← 0;
    while Q_E ≠ φ and i < T do
        i ← i + 1;
        e ← Pop(Q_E);
        for n_e ∈ N(e) do
            if Sim(n_e, n_t) > p then
                C ← C ∪ {e};
                break;
            end
        end
    end
end
```

Algorithm 1. Candidate generation and pruning

4 Experiment

The experiment is conducted on four KBP data sets (i.e. KBP2009-KBP2012) which are taken from the Knowledge Base Population (KBP) Track [16,11]. The data sets share the same track knowledge base which is derived from Wikipedia and contains 818,741 entities. We use KBP2009 and KBP2010 as the training and development data and KBP2011 and KBP2012 as the test data.

In the KBP-EL evaluation, the input is a set of queries. Each query consists of a target name mention and a context document. The output is the target entity ID in the knowledge base or NIL if the target entity is absent in the knowledge base. The number of queries/NIL-answer queries for each data set is: KBP2009: 3904/2229, KBP2010: 2250/1230, KBP2011: 2250/1126, KBP2012: 2250/1049.

Our experiments include two parts. The first part evaluates the recall and averaged candidate set size. The recall is the percentage of the non-NIL queries for which the candidate set covers the referent entity. The second part evaluates the final EL system performance, including the micro-averaged accuracy (percentage of queries linked correctly) and the averaged runtime cost per query.

[5] In this work we set the candidate set volume threshold 30 and the similarity threshold 0.6.

4.1 Evaluation on Recall

Here we compare our context based approach: CBCG with the baseline, directly matching in NEM: DMatch. Table 2 shows the recall and the averaged candidate number per query of the candidate generators. From this table, we can see that the recall of CBCG outperforms DMatch and can achieve higher than 93% on each of the data sets. On KBP2011 and KBP2012, the recall of CBCG outperforms DMatch by 15.6% and 5.2% respectively. On the other hand, the number of the candidates of CBCG only 22.5% and 9.5% of DMatch on KBP2011 and KBP2012 respectively. Few literature has reported both of the recall and the averaged candidate number. The Literature [6] reported their candidate generation recall was 0.878 and the averaged candidate number was 7.2 on KBP2009. Our approach outperforms the recall by 5.3% achieves a comparable candidate number on the same data set.

Table 2. Candidate generation recall and averaged candidate number on KBP data sets

Data Set	KBP2009	KBP2010	KBP2011	KBP2012
Recall				
DMatch	0.906	0.900	0.807	0.883
CBCG	0.931	0.964	0.963	0.935
Averaged Candidate Number				
DMatch	24.6	28.5	38.3	132.3
CBCG	8.0	8.5	8.6	12.6

The CBCG can be broken down into the following strategies:

DMatch: Directly match the target name in NEM

AcroExp: Add acronym expansion into the potential name set

LongName: Add longer co-reference of the target name into the potential name set

ShortName: Add shorter co-reference of the target name into the potential name set

fByFreq: filter by candidate frequency

fBySim: filter by candidate similarity with the target name

We add the strategies into the generator in turn to evaluate their contributions. Figure 2 shows that, directly matching the target name in NEM results in a large number of candidates. Using AcroExp, LongName and ShortName strategies, the recall will be improved. Using LongName and fByFreq, the averaged number of the candidates will be reduced significantly. Using all of these strategies, we can obtain balanced candidate sets with high recall and small size.

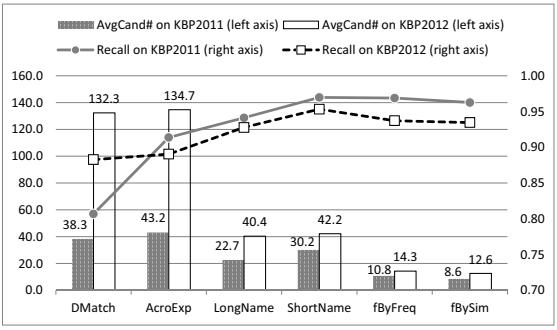


Fig. 2. Recall and averaged candidate number by different strategies

4.2 Evaluation on Accuracy

We use three candidate ranking models to evaluate the final performance of the EL system. The ranking is based on the results of our candidates generation. The ranking models are: (1) The vector space model based on cosine similarity between the candidate and the context of the target name: VSM [4,24]; (2) The machine learning model based on list wise learning to rank: ListNet [27]; and (3) The language model: LM [7]. VSM method is a simple but effective ranking model. ListNet is a state-of-the-art ranking model. LM is also a state-of-the-art ranking method but is time and memory consuming. The system output NIL if the candidate set is empty or the top ranked entity is absent from the track knowledge base. We compare the systems with the top 3 systems in the KBP evaluation. VSM+DMatch, ListNet+DMatch and VSM+CBCG, ListNet+CBCG are based on the candidate set of the DMatch baseline and the CBCG respectively. The baseline (DMatch) generated so many candidates that the LM model ran out of memory in our machine. So currently we can not provide the result of the LM model based on DMatch.

From Table 3 we can see that, the accuracies of VSM+CBCG and ListNet+CBCG are significantly higher than their DMatch versions (improved by 5.4%-11.4%) respectively, and the accuracy of LM+CBCG outperforms the best systems in the evaluations by 2.2% and 3.4% on KBP2011 and KBP2012 respectively.

4.3 Evaluation on Efficiency

Figure 3 shows the query numbers in different candidate number (i.e. polysemy) ranges on KBP2011 and KBP2012. From this figure, we can see that on KBP2011 nearly a half of the queries have more than 16 candidates, and on KBP2012 over a half of the queries have more than 64 candidates. *Jackson* is the most polysemous target name in the data sets, which has 865 candidates. From Table 2 we can see that the averaged candidate number on KBP2012 is up to 132.3. Such a big number of candidate is impractical for online applications. So the candidate pruning is essential for the candidate generation.

Table 3. EL accuracy on KBP2011 and KBP2012

Data Set	KBP2011	KBP2012
Sys1	0.863	0.766
Sys2	0.861	0.757
Sys3	0.790	0.755
VSM+DMatch	0.689	0.558
VSM+CBCG	0.777	0.672
ListNet+DMatch	0.690	0.622
ListNet+CBCG	0.786	0.676
LM+CBCG	0.885	0.800

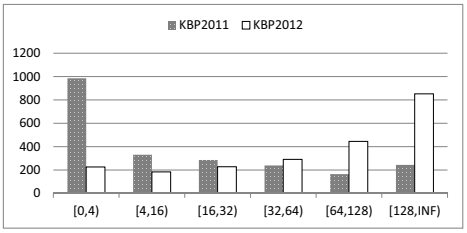
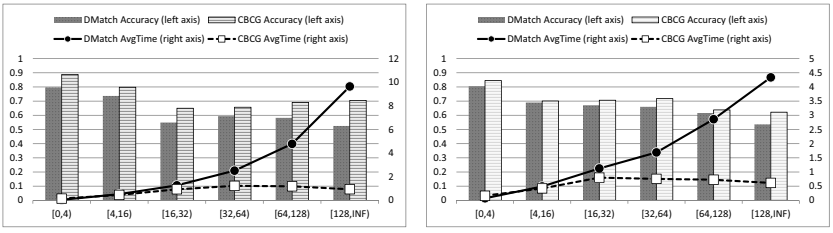


Fig. 3. Query numbers in different candidate number range

We use ListNet to evaluate the candidate generation efficiency. Figure 4 shows the runtime cost and the accuracy of the ListNet model on the baseline candidate set (DMatch) and the proposed candidate set (CBCG). From Figure 4 we can see that, the runtime increases significantly for the DMatch and keeps steady for the CBCG. By leveraging the proposed candidate set, the accuracies are improved in all candidate number ranges. For the most polysemous targets, the accuracy is improved by 18.0% and 8.6% and the runtime is saved by 90.2% and 85.8% on KBP2011 and KBP2012 respectively. In total, the accuracy is improved by 9.6% and 5.4% and the runtime of the system is saved by 70.3% and 76.6%.



(a) KBP2011 (b) KBP2012

Fig. 4. Accuracy and averaged time cost per query (seconds) of ListNet based on the DMatch and the CBCG in different polysemy ranges on KBP2011 and KBP2012

5 Conclusion

Candidate generation is essential for the EL task. The candidate number for the target names may be very large. Generating small candidate set under the premise of ensuring high recall is critical for the applications of the EL systems. In this paper we propose a novel candidate generation approach. This approach combines several strategies to balance the recall and the size of the candidate set. Experimental results on benchmark data set shows that our candidate generation can significantly improve the EL system performances on recall, accuracy and efficiency over the baseline. On the KBP2011 and KBP2012 data sets, the recall is improved by 15.6% and 5.2%, the accuracy is improved by 5.4%-11.4%, the system runtime is saved by 70.3% and 76.6%, and the highest accuracy in the evaluation is improved by 2.2% and 5.4% respectively. For the most polysemous target names on KBP2011 and KBP2012, the accuracy improvement achieves 18.0% and 8.6%, and the runtime is saved by 90.2% and 85.8% respectively.

Acknowledgments. This work was supported by National Natural Science Foundation of China (NSFC) via grant 61273321, 61073126, 61133012 and the National 863 Leading Technology Research Project via grant 2012AA011102 and the National Science and Technology Support Program via grant 2011BAH11B03.

References

1. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.G.: DBpedia: A nucleus for a web of open data. In: Aberer, K., et al. (eds.) ASWC/ISWC 2007. LNCS, vol. 4825, pp. 722–735. Springer, Heidelberg (2007)
2. Bunescu, R.C., Pasca, M.: Using encyclopedic knowledge for named entity disambiguation. In: EACL (2006)
3. Cao, Z., Qin, T., Liu, T.-Y., Tsai, M.-F., Li, H.: Learning to rank: from pairwise approach to listwise approach. In: Proceedings of the 24th International Conference on Machine Learning (2007)
4. Cucerzan, S.: Large-scale named entity disambiguation based on Wikipedia data. In: Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (2007)
5. Dredze, M., McNamee, P., Rao, D., Gerber, A., Finin, T.: Entity disambiguation for knowledge base population. In: Proceedings of the 23rd International Conference on Computational Linguistics (2010)
6. Hachey, B., Radford, W., Nothman, J., Honnibal, M., Curran, J.R.: Evaluating entity linking with wikipedia. Artificial Intelligence 194, 130–150 (2013)
7. Han, X., Sun, L.: A generative entity-mention model for linking entities with knowledge base. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Techologies (2011)
8. Han, X., Sun, L., Zhao, J.: Collective entity linking in web text: a graph-based method. In: Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information (2011)

9. Han, X., Zhao, J.: Named entity disambiguation by leveraging wikipedia semantic knowledge. In: Proceeding of the 18th ACM Conference on Information and Knowledge Management, CIKM 2009 (2009)
10. Ide, N., Véronis, J.: Introduction to the special issue on word sense disambiguation: the state of the art. Comput. Linguist. 24(1), 2–40 (1998)
11. Ji, H., Grishman, R.: Knowledge base population: Successful approaches and challenges. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (2011)
12. Kataria, S.S., Kumar, K.S., Rastogi, R.R., Sen, P., Sengamedu, S.H.: Entity disambiguation with hierarchical topic models. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2011)
13. Kulkarni, S., Singh, A., Ramakrishnan, G., Chakrabarti, S.: Collective annotation of wikipedia entities in web text. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2009)
14. Lehmann, J., Monahan, S., Nezda, L., Jung, A., Shi, Y.: Lcc approaches to knowledge base population at TAC 2010. In: Proceedings of the Text Analysis Conference (2010)
15. McCarthy, D.: Word sense disambiguation: An overview. Language and Linguistics Compass 3(2), 537–558 (2009)
16. McNamee, P., Dang, H.: Overview of the tac 2009 knowledge base population track. In: Proceedings of the Second Text Analysis Conference, TAC 2009 (2009)
17. Mihalcea, R., Csomai, A.: Wikify!: linking documents to encyclopedic knowledge. In: Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management, CIKM 2007 (2007)
18. Miller, G.A., Beckwith, R., Fellbaum, C., Gross, D., Miller, K.J.: Introduction to WordNet: An On-line Lexical Database*. Int. J. Lexicography 3, 235–244 (1990)
19. Milne, D., Witten, I.H.: Learning to link with wikipedia. In: Proceeding of the 17th ACM Conference on Information and Knowledge Management, CIKM 2008 (2008)
20. Navigli, R.: Word sense disambiguation: A survey. ACM Comput. Surv. 41, 1–69 (2009)
21. Ratinov, L., Roth, D., Downey, D., Anderson, M.: Local and global algorithms for disambiguation to wikipedia. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (2011)
22. Sen, P.: Collective context-aware topic models for entity disambiguation. In: Proceedings of the 21st International Conference on World Wide Web (2012)
23. Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: a core of semantic knowledge. In: Proceedings of the 16th International Conference on World Wide Web (2007)
24. Varma, V., Bharat, V., Kovelamudi, S., Bysani, P., Santhosh, G.S.K., Kiran Kumar, N., Reddy, K., Kumar, K., Maganti, N.: IIIT hyderabad at TAC 2009. In: Proceedings of the Second Text Analysis Conference, TAC 2009 (2009)
25. Xia, F., Liu, T.-Y., Wang, J., Zhang, W., Li, H.: Listwise approach to learning to rank: theory and algorithm. In: Proceedings of the 25th International Conference on Machine Learning (2008)
26. Zhang, W., Sim, Y.C., Su, J., Tan, C.L.: Entity linking with effective acronym expansion, instance selection, and topic modeling. In: Proceedings of the 22nd International Joint Conference on Artificial Intelligence, IJCAI 2011, Barcelona, Catalonia, Spain, July 16-22 (2011)
27. Zheng, Z., Li, F., Huang, M., Zhu, X.: Learning to link entities with knowledge base. In: Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics (2010)