International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169
Volume: 5 Issue: 6 78 – 86

Graph Based Disambiguation of Named Entities using Linked Data

Arpa H. Mirani
Computer Science Department
Visvesvaraya National Institute of Technology
Nagpur, India
miraniarpa@gmail.com

Mansi A. Radke
Computer Science Department
Visvesvaraya National Institute of Technology
Nagpur, India
mansiaradke@gmail.com

Abstract— Identifying entities such as people, organizations, songs, or places in natural language texts is needful for semantic search, machine translation, and information extraction. A key challenge is the ambiguity of entity names, requiring robust methods to disambiguate names to the entities registered in a knowledge base. Several approaches aim to tackle this problem, they still achieve poor accuracy. We address this drawback by presenting a novel knowledge-base-agnostic approach for named entity disambiguation. Our approach includes the HITS algorithm combined with label expansion strategies and string similarity measure like the n-gram similarity. Based on this combination, we can efficiently detect the correct URIs for a given set of named entities within an input text.

Keywords-*Disambiguation, HITS Algorithm, Linked Data*

*****

I. INTRODUCTION

In natural language processing, Disambiguation (also called word sense disambiguation or text disambiguation) is the act of interpreting an author's intended use of a word that has multiple meanings or spellings. It is the activity of resolving conflicts which arise when an article title is ambiguous, mostly because it refers to more than one subject.

A disambiguation technique is called graph based disambiguation when it makes use of a graph to resolve the ambiguity between identified labels in a text. A graph is formed using the potential candidates and then the correct candidate is identified using certain measures on the graph.

Named entity disambiguation (NED) is the task of determining the identity of entities mentioned in text. For example, given the sentence "Paris is the capital of France", the idea is to determine that "Paris" refers to the city of Paris and not to Paris Hilton or any other entity that could be referred as "Paris". In this example, an NER tool would identify the entities Paris and France. A high-quality DBpedia-based named entity disambiguation (NED) approach should use these already recognized named entities and map the strings Paris and France to the resources dbr:Paris and dbr:France [17].

Named entity mentions can be highly ambiguous; any entity linking method must address this inherent ambiguity. Various approaches to tackle this problem have been tried till date. However these approaches suffer from two major drawbacks: First, they perform poorly on Web documents as they contain resources from different domains within a narrow context. An accurate processing of Web data is important for the implementation of the Web of Data. Well-known approaches such as Spotlight [11] and TagMe 2 [18] have been designed to work on a particular knowledge base. However, Web data contains resources from many different domains. Hence, NED approaches have to be designed in such a way that they are agnostic of the underlying knowledge base. Second, most state-of-the-art approaches rely on exhaustive data mining methods [3,14] or algorithms with non-polynomial time complexity. However, given the large number of entities that must be disambiguated when processing Web documents, scalable NED approaches are of central importance to realize the Semantic Web vision.

II. LITERATURE REVIEW

NED in web documents is well studied in literature. Several approaches use Wikipedia or a KB derived from Wikipedia (like DBpedia and YAGO) as entity collection to look-up for the appropriate entity for a mention.

One of the earliest approaches was proposed by Bunescu et al.[2]. They developed a named entity disambiguation technique that performs disambiguation in two steps. First, it finds whether a proper name refers to a named entity present in the dictionary (detection). Second, it disambiguates between several named entities which can be referred by the same proper name (disambiguation). Moreover, the authors developed a similarity measure which compared the context of a mention with the Wikipedia categories of an entity.

Cucerzan[3] proposed a technique for disambiguating named entities using information retrieved from Wikipedia and the web. The technique uses the data associated with the known surface forms within a document and all their entity disambiguations so as to obtain a greater similarity between the context data obtained for the candidate entities and the information in the context of the document.

The importance of coherence measure between two entities in disambiguation was introduced by Kulkarni et al. [4]. In the same way, Hoffart et al. [5] combined three measures: the prior probability of an entity being mentioned, the similarity between the contexts of a mention and a candidate entity, as well as the coherence among candidate entities for all mentions together. AIDA [6] is a system built on Hoffart's [5] approach.

Ad-hoc (entity oriented) NED shows yet another direction in NED research. Ad-hoc entities are not a part of a knowledge base like DBpedia, YAGO or Freebase. Instead of using a KB, given the candidate labels of all the target entities, entity oriented disambiguation techniques determine which ones are correct mentions of a target entity. Srinivasan et al. proposed a cross document person name disambiguation technique that groups documents so that each group contains all and only those documents addressing the same person. They introduced

characteristics based on topic models and also document-level entity profiles of information that are created for each ambiguous person in the whole document.

Wang et al. introduced disambiguation strategies that require no knowledge about the entity mentions except their names. They proposed a graph-based model called MentionRank to affect the uniformity constraint and disambiguate the candidate labels collectively in the document. Holding the uniformity constraint of the entities is done in three ways: context similarity, co-mentioned entities, and cross-document, cross- entity interdependence.

Another approach is DBpedia Spotlight [11], a framework for annotating and disambiguating Linked Data Resources in random texts. In comparison to other tools, Spotlight is able to disambiguate all classes of the DBpedia ontology. Moreover, it is well-known in the Linked Data community and used in various projects showing its wide-spread adoption. It is based on a vector-space model and cosine similarity.

In 2012, Ferragina et al. published a reviewed version of their disambiguation technique called TagMe 2. The authors suggest that it is meant for smaller texts, i.e., containing around 30 terms. TagMe 2 is based on an anchor listing (<a> tags on Wikipedia pages with a certain frequency), a page listing containing all original Wikipedia pages and an in-link graph. First, TagMe 2 identifies named entities by matching terms with the anchor listing and then disambiguates the match using the in-link graph and the page listing via a collective result of identified anchors. Lastly, the technique eliminates identified named entities which are not coherent to the rest of the named entities in the input text.

In 2014, Babelfy [13] was proposed. It is based on arbitrary passes and densest sub-graph algorithms to tackle NED and is evaluated with six datasets, one of them used AIDA dataset. In contrast to proposed approach, Babelfy differentiates between word sense disambiguation and entity linking.

III. PROPOSED APPROACH

A. Objective

The objective of this project is to recognize correct resources from a Knowledge base K for n a-priori determined named entities N1; : : : ;Nn obtained from a given input text T. In general, there can be several candidate resources from a given knowledge base K for a given entity Ni. Out of these candidate resources only one candidate resource can be the required resource. Our aim is to find out that resource correctly. In the result set we get N such resources where each resource stands for an entity from the given input text.

Similarity functions are applied to calculate similarity between candidate resources and named entity. Let ψ be the similarity function, e.g., string similarity. The coherence function calculates the similarity of the knowledge base K and candidate resources to ensure dependable relevance. Let Φ be the coherence function. It is implemented by the HITS algorithm to compute the closest entities. Given this formal model, the objective is to find the result R with

$$R = arg\,max(\,\psi(candidate\ resource, named\ entity) + \phi(candidate\ resource, K))$$

For the purpose of scalability, we compute the result by using an upper bound of $\Theta(k\ .|V\ |2)$ on the HITS algorithm, where k is the number of iterations and |V| is the number of nodes in the graph. Moreover, using HITS has other advantages like

1) scalability,
2) well explored behavior and
3) the ability to analyze semantic authority.

Our approach for NED consists of three stages.We first retrieve all named entities from the given input text using a named entity recognition function (e.g., Stanford NER[7]). After that, we detect candidates for each of the extracted named entities. We apply several heuristics at this stage and make use of known surface forms [11] for resources from the underlying knowledge base. We then use this list of candidates generated in the previous step to form a disambiguation graph. For this we make use of a graph search algorithm(e.g. Breadth First Search) on the underlying knowledge base to extract context information. Finally, in the last step, we implement the HITS algorithm on the obtained disambiguation graph to find most appropriate candidates for the extracted named entities in our input text. The candidates with the highest authority values are the correct resources. All stages in our project have a polynomial time complexity, leading to overall project also being polynomial in time complexity. We present each of the steps of our project in more detail below.

B. String normalization and expansion

An index is created, it is used to search candidates for the entities identified by Stanford NER. We get a list of possible candidates from the index. After searching the index, a string normalization function and an expansion policy is applied to the input text. The string normalization is a technique to remove plural and genitive forms(e.g. apostrophe), eliminating common bound forms like postfixes for corporate labels and not including candidates with time information(years, dates, etc.) within their label. For example, the genitive India's is transformed into India, the postfix of Reliance Ltd. is reduced to Reliance and the time information of Mumbai 2015 is ignored.

Named entities are usually mentioned in their full length only the first time they appear in news and web pages, whereas the other repetitive mentions only have a substring of the actual mention due to the concise nature of most news articles. For example, a text mentioning Narendra Modi's visit to the United States of America will mostly contain Narendra Modi in the first mention of him and use strings as Modi later in the same text as the readers know whom is it referring to from the first mention of him. According to this perception, we map each named entity label (e.g., Modi) which is a substring of another named entity label that was recognized previously (e.g., Narendra Modi) to the same resource , i.e., dbr: Narendra Modi. We choose the shortest expansion if there are several expansions in the input text. This expansion policy is an approach to co-reference resolution within the same document, it is time-efficient and plays a very important role in dealing with text from the web.

The lists of candidates are then filtered out and most relevant candidates for each named entity are found out.

C. Filtering the candidates

For this filtration step, measures such as string similarity, domain check are used. We compare the named entity and each of the candidates extracted in the previous step using n-gram similarity measure. We have used trigram similarity, where the value of n is 3. The domain check measure is used to narrow down the search space, that is to eliminate the candidates which do not belong to any of the domains specified for named entity recognition, because these candidates can simply not be the

correct resource for the named entity since their domains are different. We have limited our technique to named entities which fall under the categories such as person, organization and place. So candidates other than these domains are eliminated in this step. These categories can be changed by the user according to his purposes. The types are mentioned in Table 1 for DBpedia and YAGO2 knowledge bases that we have used. This technique is added to decrease the number of candidates which in turn increases the accuracy.

Class	rdf:type
DBpedia Person	dbo:Person, foaf:Person
DBpedia Organization	dbo:Organization, dbo:WrittenWork (e.g., Journals)
DBpedia Place	dbo:Place, yago:YagoGeoEntity
YAGO2 Person	yago:yagoLegalActor
YAGO2 Organization	yago:yagoLegalActor, yago:wordnet_exchange_111409538 (e.g., NASDAQ)
YAGO2 Place	yago:YagoGeoEntity

Table 1: DBpedia and YAGO2 classes used for disambiguation classes.

The candidate detection and filtration approach explained above is summarized in Algorithm 1.

Algorithm 1: Searching candidates for a label.(courtesy [1])

```
Data: label of a certain named entity N_i, σ trigram similarity threshold
Result: C candidates found
C← Ø;
label ← normalize(label);
label ← expand(label);
-C ← searchIndex(label);
for c ϵ -C do
    if ~c.matches([0-9]+) then
        if trigramSimilarity(c, label) ≥ σ then
            if fitDomain(c) then
                C ← C ∪ c;
```

So after this step we get a filtered list of candidates out of which one maybe our required entity.

D. *Construction of graph*

Once we have a filtered list of candidates for each named entity we start constructing a disambiguation graph. For this we use search algorithm like breadth first search which is applied on the knowledge base to get context of the candidate nodes. Our knowledge base is in the form of a directed graph, where there are edges between the resources which are the nodes and (subject; predicate; object) is an RDF triple in the KB. In the beginning our graph contains all the candidates as its nodes and set of edges is set to null. To expand the graph we search for contextually related nodes to our candidate node for each named entity one by one. The extension of the graph is defined below,

Vi+1=Vi ∪ {y|∃x which is a node in the disambiguation graphΛ (subject, object) is an edgein the KB}

Ei+1 = {(subject, object) is an edge in the KB | x, y are the added nodes}

We iterate the extension step d times on the initial graph to obtain the final disambiguation graph. This depth d for the number of iterations can be set according to the required accuracy.

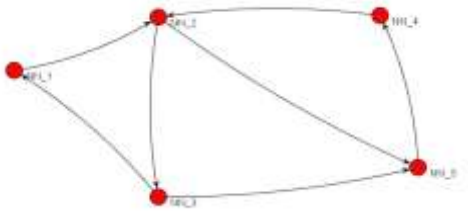

Fig. 1: A simple directed graph using JUNG2

For construction of graph we have used the JUNG framework in java as shown in fig. 1.

E. *HITS algorithm*

Hyperlink-Induced Topic Search is an algorithm developed by Jon Kleinberg. It is a link analysis algorithm that ranks web pages. The idea about Hubs and Authorities came up from a perception of the formation of web pages when the Internet was originally being created; that is, some web pages, called as hubs, performed the task of large directories which were not actually reliable for the information that they contained, but were used as collections of a wide prospectus of information that helped users to reach authoritative pages. So we can conclude that, a good hub constituted a page that directed to several other pages, and a good authority constituted a page that was directed to by several different hubs.

The algorithm therefore gives two scores for every page: authority score, which rates the importance of the matter of the page, and hub score, which rates the importance of its links to other pages.

In the HITS algorithm, the starting thing is to extract the most relevant pages to the given query. This set is referred as the root set and we can get it by taking the top pages returned by a text-based search algorithm. A base set is generated by extending the root set to add all the web pages that are connected from it and the pages that link to it. The web pages in the base set along with all the hyperlinks among those pages form a focused sub-graph. The HITS calculation is done only on this focused sub-graph.

Authority and hub values are calculated in terms of each another in a mutual recursion. An authority value is calculated as the sum of the normalized hub values that point to that page. A hub value is the sum of the scaled authority values of the pages it points to.

The algorithm performs a number of iterations, each consisting of two basic steps:

Authority Update: Update each node's Authority score which is equal to the sum of the Hub Scores of each node that links to it. That is, a node gets a high authority score by being pointed from pages that are identified as Hubs for information.

Hub Update: Update each node's Hub Score which is equal

to the sum of the Authority Scores of each node that it links to. That is, a node is given a high hub score by pointing to nodes that are considered to be authorities on the subject.

HITS, like PageRank method, is an iterative algorithm based on the connections of the documents on the web. But it is query dependent, that is, the search terms affect the Hub and Authority scores obtained from the link analysis.

Using the HITS algorithm we compute authority scores x_a; y_a and hub scores x_h; y_h for all x; y which are nodes in the disambiguation graph. We initialize the authority and hub scoresand afterwards repeat the equationk times as given below:

$$\forall x \in V_d, x_a = x_h = \frac{1}{|V_d|}$$

$$x_a \leftarrow \sum_{(y,x)\epsilon E_d} y_h \, , y_h \leftarrow \sum_{(y,x)\epsilon E_d} x_a$$

We have chosen k according to Kleinberg [10], i.e., the value of k is set to 20, which are sufficient to achieve convergence in most cases. Then we find the candidate with the highest authority value among the list of candidates as correct resource for a given named entity. The whole procedure is presented in Algorithm 2.

Algorithm 2: Disambiguation Algorithm based on HITS and Linked Data.(courtesy [1])

Data: N = {N_1; N_2 : : : N_n} named entities, σ trigramsimilarity threshold, d depth, k number of iterations
Result: C = {C_1; C_2 : : : C_n} identified candidates for named entities

```
E ← Ø;
V ← insertCandidates(N, σ);
G ← (V, E);
G ← breadthFirstSearch(G; d);
HITS(G(V,E),k);
sortAccordingToAuthority(V);
  for Ni ϵ N do
      for v ϵ V do
          if v is a candidate for Ni then
              store(Ni, v);
              break;
```

To improve accuracy we have added a few more steps.

F. Using the Wikipedia Disambiguation Pages

Following are the categories of Wikipedia pages :
Article titles (Title) : The title of the article. The first letter of Wikipedia titles is case-insensitive and by default given in the uppercase form. For the articles having the special lowercase starting letter (like in gzip, iPod), we retrieve this alias with its first alphabet lowercased.
Redirect titles (Redirect): Wikipedia provides a redirect technique to automatically forward a user from non-authorized titles — such as variant or wrong spellings, short forms, different language titles etc. — to the relevant article. e.g. for articles with lowercase title, if the redirect title begins with the first word of the authorized title, its first letter is also lowercased (e.g., IPods becomes iPods).
Disambiguation page titles (DABTitle): Disambiguation pages are used to list the articles which may be represented to by an ambiguous title. The heading of a disambiguation page (e.g., an abbreviation or a surname) is thus taken as another name of the pages it disambiguates. Disambiguation pages commonly consist of more than one list, where each item of the list links to a candidate referent of the disambiguated name. However, such links are not limited only to candidates, so we only consider links that appear at the top of a list item. All descendants of the Disambiguation pages category are referred as disambiguation pages. We have shown an example of a disambiguation page below.
We are already using the redirect pages to get the correct candidate even if there is a spelling mistake or typographical error in the label occurring in the given web page.

Fig. 2: Wikipedia Disambiguation page for Kashmir

To add candidates that have been missed out due to the high trigram similarity threshold requirement, we have made use of the disambiguation pages. The trigram similarity threshold for these mentions is slightly low to qualify as a candidate for the label. This increases accuracy and decreases the chance of NIL entity. The algorithm for this is mentioned below.

```
for each entity e1
  if disambiguation page is present
        for each resource in disambiguation page
              check for trigram similarity
                    if similarity > threshold2
                          add the resource in candidate list
                    end if
        end for
  end if
end
```

Algorithm 3: Use of Disambiguation pages

G. Context Similarity

Usually a text revolves around some context. Even if the whole text does not point towards the same topic, the labels in the same sentence or within a threshold distance have context common between them. We have made use of this fact to increase the accuracy of this project.
If a possible candidate to an entity is found in the context of another entity, we add that as a candidate for the particular entity if it is not already present in the list of candidates. Due to this the missed out candidate which has a greater chance of

being the correct candidate is added in the list. The algorithm for the same is mentioned below.

```
for each entity e1
  if superstring s of other entity e2 is present in context of e1
        if s not present in candidate list of e2
              add that resource as candidate of e2
        end if
  end if
end
```

Algorithm 4: Context Similarity

H. Modified HITS algorithm

The idea behind the HITS algorithm can be understood from the following qualitative recursive definition: "A good authority is the one which is pointed to by many good hubs, and a good hub is the one which points to many good authorities". Thus, the value of some page p as an authority (obtained by the authority weight of page p) depends on the value of the pages that point to p as hubs (obtained by the hub weight of the pages), and vice versa. Kleinberg proposed to compute the hub and authority values using the addition operation. The authority weight of a page is obtained by the sum of the hub weights of the pages that link to p, and the hub weight of the page is defined to be the sum of the authority weights of the pages that are linked to by it. Two properties can be inferred from this definition. First, it is symmetric, because both hub and authority scores are calculated in the same way. If we change the direction of the edges, then authority and hub weights are interchanged. Second, while computing the hub weight of some page p, the authority weights of the pages that are linked to by page p are all considered equal (similarly while computing the authority weights).

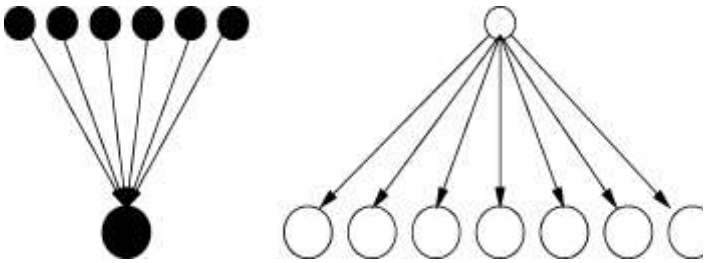

Fig.3: A bad example for HITS algorithm.

These two things may sometimes lead to unexpected results. Consider, for example, the graph in Figure 3. In this graph, there are two components. The a part of the figure has a single authority linked to by a large number of hubs. The b part of the figure has a single hub that links to a large number of authorities. When the number of part b authorities is larger than the number of part a hubs, the HITS algorithm will allocate all authority weight to the part b authorities, while giving zero weight to the part a authority. The reason for this is that the part b hub is declared to be the best hub, thus causing the part b authorities to receive more weight. However, logically the part a authority is better than the part b authorities and should be ranked higher. In this example, the two properties of the HITS algorithm result in an unexpected result. Equality means that all authority weights of the nodes that are linked to by a hub serve equally to the hub weight of the node. As a result, quantity becomes quality. The hub weight of the part b hub increases simply because it points to many weak authorities. This leads us to question the definition of the hub weight and thus, the other symmetric nature of HITS. Symmetry assumes that hubs and authorities are qualitatively the same. However, there is a difference between the two. For example, logically a node with high in-degree is likely to be a good authority while, a node with high out-degree is not necessarily a good hub. If it was true, then it would be very simple to increase the hub quality of a page, just by adding links to random pages. Thus it means that we should treat hubs and authorities differently.

Borodin proposed a modification of the HITS algorithm to solve this problem. The Hub-Averaging algorithm updates the authority weights like the HITS algorithm, but it sets the hub weight of some node i to the average authority weight of the authorities pointed to by hub i. Thus, for some node i, we have The logic behind the HUBAVG algorithm is that a good hub should point only to good authorities, rather than to both good and bad authorities. Note that in the example in Figure 3, HUBAVG assigns the same weight to both part a and part b hubs, and it identifies the part a authority as the best authority. The HUBAVG algorithm is shown below.

Algorithm 5: The HUBAVG (courtesy[19])

Initialize authority weights to 1
Repeat until the weights converge:
For every hub i ϵ H

$$hi = \frac{1}{|F(i)|}\Sigma aj$$

For every authority i ϵ A

$$ai = \Sigma hj$$

Normalize

The HUBAVG algorithm can be seen as a mixture of the HITS and SALSA algorithms. The averaging of the weights of the authorities linked to by a hub is similar to dividing the weight of a hub between the authorities it links to. Therefore, the HUBAVG algorithm performs the calculation of authority weights like the HITS algorithm and the calculation of hub weights like the SALSA algorithm. This absence of symmetry in the calculation of hubs and authorities is influenced by the qualitative difference between hubs and authorities explained previously.

I. Pre-processing step before HITS

As we know the data in the world is large today and there can be a number of candidates for a single entity. So the disambiguation graph that is formed is very large in size. To reduce the graph size we have added a pre-processing step before passing the graph to HITS algorithm.

The idea of this step is to remove those nodes from the graph which will not affect the output largely and are far away

 ISSN: 2321-8169
78 – 86

from the candidates in context as well as depth in the graph. After applying this step many nodes are removed without affecting the result and thus the HITS algorithm has to be applied on a comparatively smaller graph which in turn increases the accuracy.

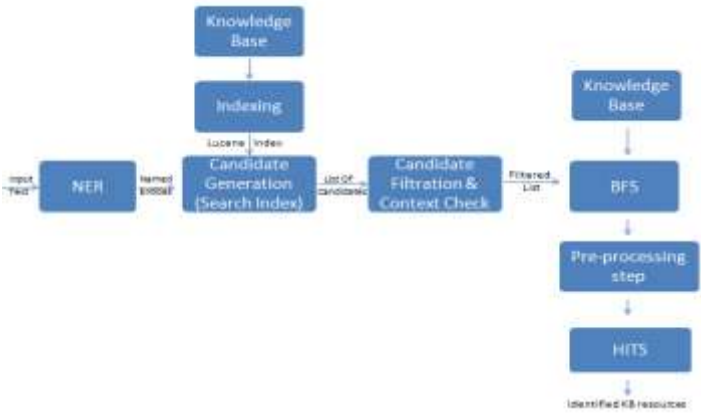

Fig. 4: Flow Diagram of the Proposed Approach

To explain the proposed approach with an example, we use a input text shown in fig 5.

During his standout career at Bulls, Jordan also acts in the movie Space Jam.

Fig. 5: Input text

J. Explanation using an example

Step 1: Named Entity Recognition (NER)

The named entities are identified from the given input text shown in fig. 5.
The output of this step is shown in fig. 6

During his standout career at <entity>Bulls</entity>, <entity>Jordan</entity> also acts in the movie <entity>Space Jam</entity>.

Fig. 6: Output after NER

Step 2: Candidate Generation
The Lucene index is searched and candidates are found for each of the identified label in the text.
The screenshots for candidates of each of the entities are shown below in fig 7,8,9.

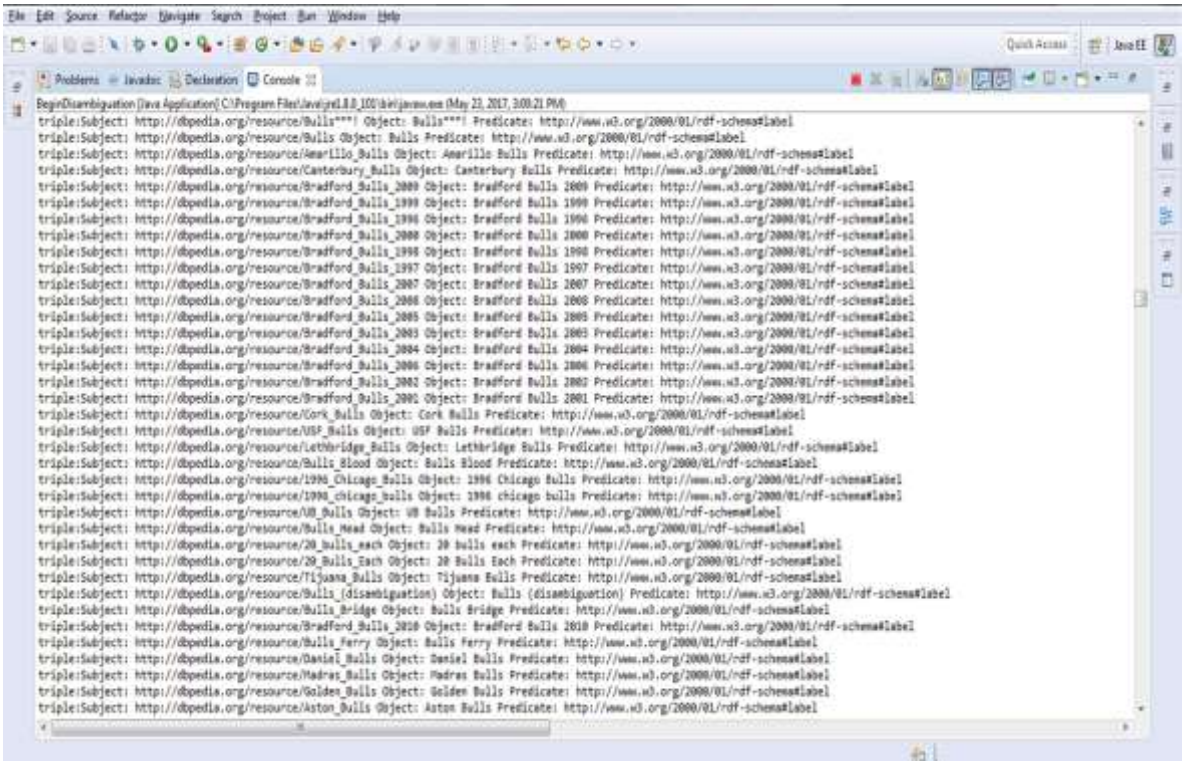

Fig.7: Candidates for entity Bulls

 ISSN: 2321-8169
78 – 86

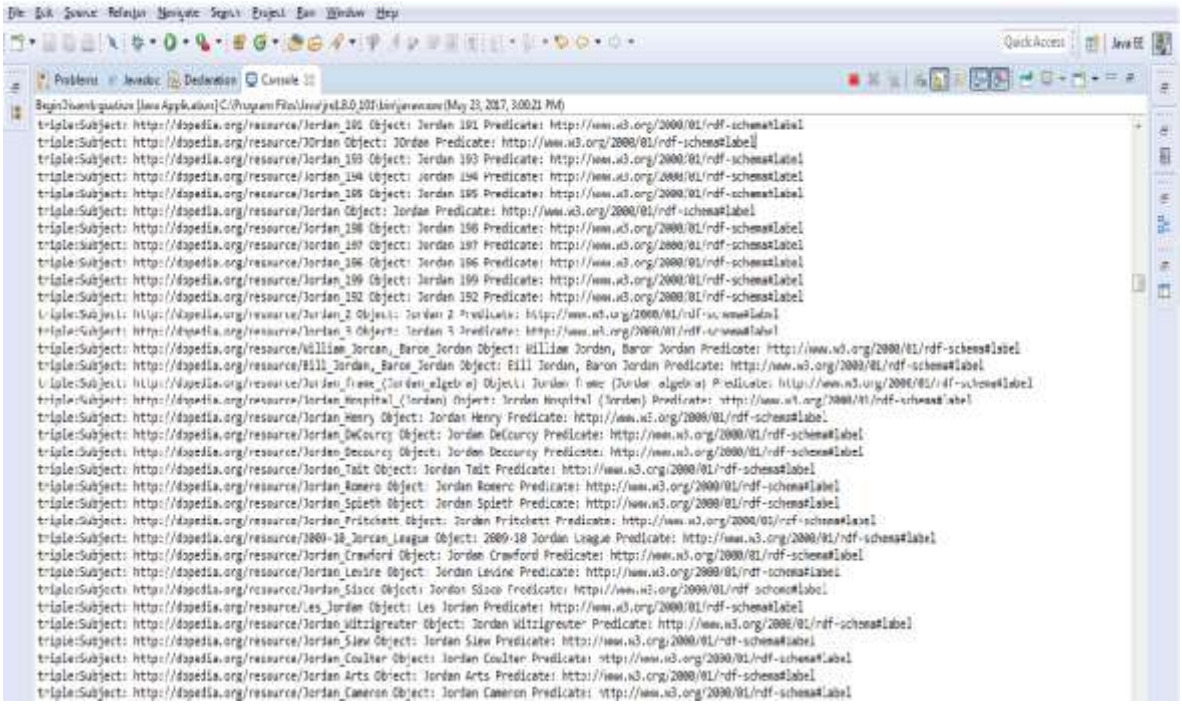

Fig. 8: Candidates for entity Jordan

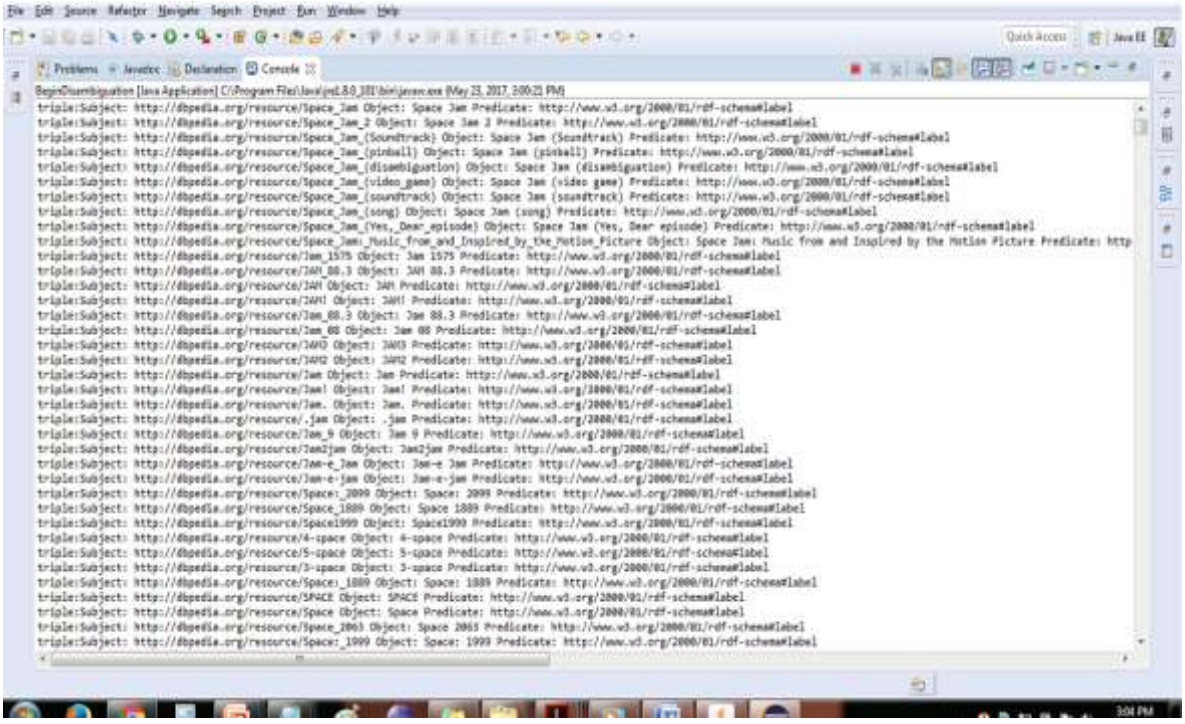

Fig. 9: Candidates for entity Space Jam

Step 3. Candidate Filtration

In this step n-gram similarity, FitDomain(), context similarity etc are applied and the final list of candidates is prepared.

The output after this step is shown in fig. 10.

```
http://dbpedia.org/resource/Michael_Jordan
http://dbpedia.org/resource/Space_Jam
no of vertices in graph: 2
level right now: 0
```

Fig. 10: Candidate Filtration output

Step 4. BFS on KB to form Disambiguation graph
The final list of candidates and KB are used and a disambiguation graph is formed.

The nodes of the graph are printed in the figure11.

```
Printing nodes in graph
http://dbpedia.org/resource/Michael_Jordan
http://dbpedia.org/resource/Joe_Pytka
http://dbpedia.org/resource/Danny_DeVito
http://dbpedia.org/resource/Michael_Chapman_(cinematographer)
http://dbpedia.org/resource/Wayne_Knight
http://dbpedia.org/resource/Herschel_Weingrod
http://dbpedia.org/resource/James_Newton_Howard
http://dbpedia.org/resource/Michael_Jordan__2
http://dbpedia.org/resource/Space_Jam
http://dbpedia.org/resource/Michael_Jordan__1
http://dbpedia.org/resource/Daniel_Goldberg_(producer)
http://dbpedia.org/resource/Warner_Bros._Family_Entertainment
http://dbpedia.org/resource/New_York
http://dbpedia.org/resource/Ivan_Reitman
http://dbpedia.org/resource/Timothy_Harris_(writer)
http://dbpedia.org/resource/Sheldon_Kahn
http://dbpedia.org/resource/Joe_Medjuck
http://dbpedia.org/resource/Shooting_guard
```

Fig. 11: Output after BFS

Step 5. Pre-processing step

The nodes which do not affect the output largely are removed and the minimized graph is shown in Fig. 12.

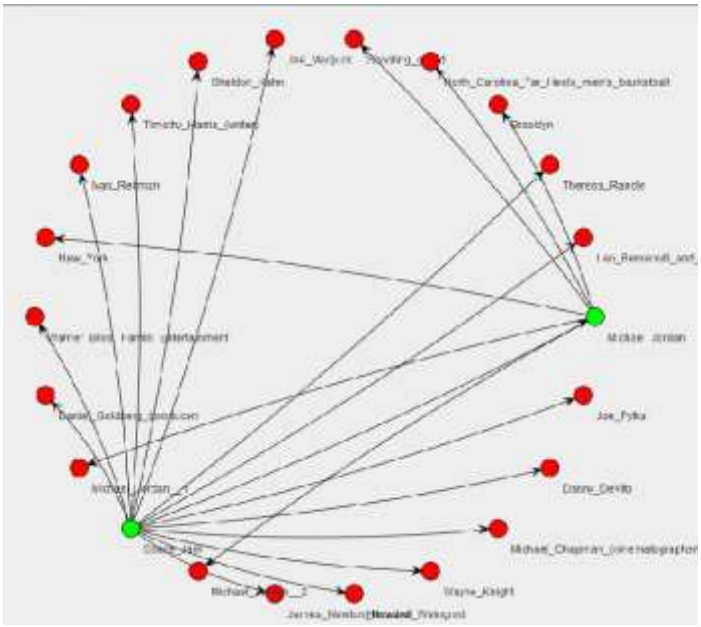

Fig. 12: Output after pre-processing step

Step 6. HITS Algorithm

Lastly, HITS algorithm is performed on the minimized graph and result set is obtained which contains the identified resources from the knowledge base for each entity in the input text.
The result set is shown in fig. 13.

```
Result:
{67=http://dbpedia.org/resource/Space_Jam,
37=http://dbpedia.org/resource/Michael_Jordan
}
```

Fig. 13: Final Output of proposed approach

IV. EXPERIMENTS AND RESULTS

For the experimental analysis, we have used the following datasets.

1. AIDA/CO-NLL-TestB : This dataset is derived from the Cornolti et al. benchmarks and begins from the evaluation of AIDA [9]. This dataset was derived from the CO-NLL 2003 shared task[15] and contains 1,393 news articles that were manually annotated. The second test part of the Cornolti et al.'s benchmark comprises of 231 documents having 19.4 entities in each document on an average.
2. AQUAINT : In this dataset, only the first mention of an entity is annotated. It consists of 50 documents which are on average longer than the AIDA/CO-NLL-TestB documents. Every document has 14.5 annotated elements on an average. The documents are from different news services, e.g. Associated Press and have been annotated using voter agreement. The dataset was created by Milne et al.

Corpus	Language	#Doc.	#Ent.	Ent./Doc.	Annotation
AIDA/CO-NLL-TestB	English	231	4458	19.40	voter agreement
AQUAINT	English	50	727	14.50	voter agreement

Table 2: Test corpora specification

Dataset	Approach	F1-measure	Precision	Recall
AIDA/CO-NLL-TestB	**TagMe 2**	0.565	0.58	0.551
	DBpedia Spotlight	0.341	0.308	0.384
	Proposed Approach	**0.636**	**0.723**	**0.568**
AQUAINT	**TagMe 2**	0.457	0.412	**0.514**
	DBpedia Spotlight	0.26	0.178	0.48
	Proposed Approach	**0.576**	**0.796**	0.451

Table 3: Performance of Proposed Approach, DBpedia Spotlight and TagMe 2 on the datasets using micro F-measure (F1).

We compared our approach with TagMe 2 and DBpedia using these datasets which were already implemented in the Cornolti et al framework. Proposed approach has used a breadth-first search with depth d = 2 and a trigram similarity with a threshold of 0.82. All the approaches that have been used above disambiguate using the English DBpedia.

V. CONCLUSION AND FUTURE WORK

We have presented a novel named entity disambiguation approach in this project. Our approach combines the Hypertext-Induced Topic Search (HITS) algorithm with label expansion strategies and string similarity measures. It is a graph based approach for named entity disambiguation. Our approach has been tested on two datasets, namely, AIDA/CONLL-TESTB

and AQUAINT datasets from the Cornolti. et al benchmark. It outperforms the state-of-the-art algorithms TagMe2, and DBpedia Spotlight while remaining quadratic in its time complexity. We achieve 0.636 F1-measure for the AIDA/CONLL-TESTB dataset and 0.576 F1-measure for the AQUAINT dataset.

For future work, following modifications can be done .

Our approach can be extended to implement a paragraph wise disambiguation policy. It performs disambiguation on whole documents. Large number of resources in the documents thus causes our approach to generate very large disambiguation graphs. The number of errors in these graphs leads to an overall low performance disambiguation. This drawback can be dealt with in future work by fitting our approach with a preprocessor which is able to extract paragraphs from input texts.

REFERENCES

[1] Usbeck, Ricardo, et al. "AGDISTIS-graph-based disambiguation of named entities using linked data." *International Semantic Web Conference*. Springer International Publishing, 2014.

[2] Bunescu, Razvan C., and Marius Pasca. "Using Encyclopedic Knowledge for Named entity Disambiguation." *Eacl*. Vol. 6. 2006.

[3] Cucerzan, Silviu. "Large-scale named entity disambiguation based on Wikipedia data." (2007).

[4] Kulkarni, Sayali, et al. "Collective annotation of Wikipedia entities in web text." *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2009.

[5] Hoffart, Johannes, et al. "Robust disambiguation of named entities in text." *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2011.

[6] Yosef, Mohamed Amir, et al. "Aida: An online tool for accurate disambiguation of named entities in text and tables." *Proceedings of the VLDB Endowment* 4.12 (2011): 1450-1453.

[7] Finkel, Jenny Rose, Trond Grenager, and Christopher Manning. "Incorporating non-local information into information extraction systems by gibbs sampling." *Proceedings of the 43rd annual meeting on association for computational linguistics*. Association for Computational Linguistics, 2005.

[8] B. Adida, I. Herman, M. Sporny, and M. Birbeck. RDFa 1.1 Primer. Technical report, World Wide Web Consortium, http://www.w3.org/TR/2012/NOTE-rdfaprimer-20120607/, June 2012.

[9] Cornolti, Marco, Paolo Ferragina, and Massimiliano Ciaramita. "A framework for benchmarking entity-annotation systems." *Proceedings of the 22nd international conference on World Wide Web*. ACM, 2013.

[10] Kleinberg, Jon M. "Authoritative sources in a hyperlinked environment." *Journal of the ACM (JACM)* 46.5 (1999): 604-632.

[11] Mendes, Pablo N., et al. "DBpedia spotlight: shedding light on the web of documents." *Proceedings of the 7th international conference on semantic systems*. ACM, 2011.

[12] Milne, David, and Ian H. Witten. "Learning to link with wikipedia." *Proceedings of the 17th ACM conference on Information and knowledge management*. ACM, 2008.

[13] Moro, Andrea, Alessandro Raganato, and Roberto Navigli. "Entity linking meets word sense disambiguation: a unified approach." *Transactions of the Association for Computational Linguistics* 2 (2014): 231-244.

[14] Ratinov, Lev, et al. "Local and global algorithms for disambiguation to wikipedia." *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics, 2011.

[15] Tjong Kim Sang, Erik F., and Fien De Meulder. "Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition." *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*. Association for Computational Linguistics, 2003.

[16] Mihalcea, Rada, and Andras Csomai. "Wikify!: linking documents to encyclopedic knowledge." *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*. ACM, 2007.

[17] Lehmann, Jens, et al. "DBpedia–a large-scale, multilingual knowledge base extracted from Wikipedia." *Semantic Web* 6.2 (2015): 167-195.

[18] Ferragina, Paolo, and Ugo Scaiella. "Fast and accurate annotation of short texts with wikipedia pages." *IEEE software* 29.1 (2012): 70-75.

[19] Borodin, Allan, et al. "Link analysis ranking: algorithms, theory, and experiments." *ACM Transactions on Internet Technology (TOIT)* 5.1 (2005): 231-297.