Exploiting Wikipedia to Identify Domain-Specific Key Terms/Phrases from a Short-Text Collection

M. Atif Qureshi[1][2], Colm O'Riordan[1], and Gabriella Pasi[2]

[1] Computational Intelligence Research Group, Information Technology,
National University of Ireland, Galway, Ireland
{muhammad.qureshi, colm.oriordan}@nuigalway.ie

[2] Information Retrieval Lab, Informatics, Systems and Communication,
University of Milan Bicocca, Milan, Italy
{muhammad.qureshi, pasi}@disco.unimib.it

Abstract. Extracting from a given document collection what we call "domain-specific" key terms/phrases is a challenging task. By "domain-specific" key terms/phrases we mean words/expressions representative of the topical areas specific to the focus of a document collection. For example, when a collection is related to academic research (i.e., its focus is related to topics dealing with academic research), the domain-specific key terms/phrases could be 'Information Retrieval', 'Marine Biology', 'Science', etc. In this contribution a technique for identifying domain-specific key terms/phrases from a collection of documents is proposed. The proposed technique works on short textual descriptions, and it makes use of the titles of Wikipedia articles and of the Wikipedia category graph. We performed some experiments over the document collection (html title text only) of eight post-graduate school Web sites of five different countries. The evaluations show promising results for the identification of domain-specific key terms/phrases.

1 Introduction

In domain-specific search applications, documents in the indexed collection cover topics which are related to the focus of the document collection. Finding domain-specific key terms/phrases (from now on referred to as "keywords") from a given collection is a significant research challenge. Despite the fact that in the literature, several approaches have been proposed to extract knowledge from a text, the goal of current knowledge extraction approaches remains limited to the identification of keywords that describe the document content independent of the focus of the considered collection from where it was drawn. Among the several approaches, the one based on classical tf-idf [18] is shown in a recent study to form a strong baseline when compared across different datasets [7].

The approaches in the literature fall into four categories: statistical learning techniques [20] [22], techniques based on latent variable topic models [3], techniques utilizing open-domain knowledge resources [13] [14], and techniques based on word graphs [21].

However, the current approaches do not focus on extracting the domain-specific keywords from a given document collection. Moreover, these approaches operate on full-text documents with the consequence of being computationally expensive. We address the problem of extracting domain-specific keywords from a given collection using short-text snippets (i.e., the text of title of a Web page) of each document. We present a novel domain-specific keyword extraction method built upon n-gram overlap between the titles of Wikipedia articles and documents (the text of titles of Web pages); the proposed method is aimed at discovering keywords related to domains from a text collection. Furthermore, our method applies a community detection algorithm (by making use of Wikipedia Category graph) with the aim of reducing the candidates to domain-specific keywords.

In order to prove the effectiveness of the proposed method we have performed experiments on a collection of academic Web sites; we show that the proposed method outperforms existing baseline algorithms i.e., classical tf-idf [18] and BM25 [16].

The rest of the paper is organized as follows. In Section 2, we describe the research background by giving an overview of the related literature. In Section 3, we discuss the underlying methodology for extraction of the domain-specific keywords using Wikipedia articles and Wikipedia category graph. In Section 4, we present some experimental evaluations along with their results. Section 5, concludes the paper with possible future directions.

2 Related Work

Several approaches have been proposed in the literature to address the problem of keyword extraction from a text. Approaches based on the tf-idf model [16] [18] are the oldest ones that incorporate the influence of the collection while estimating the relevance of keywords for each document. The research community investigated supervised learning methods [20] [22] where training data is used to provide syntactic and lexical features for keywords extraction. A more recent line of research utilizes features extracted from open-domain knowledge resources such as Wikipedia for improving the accuracy of supervised learning based keywords extraction systems [13] [14]. However, supervised learning is a laborious task and is not desirable for Web scale data.

On the other hand, an alternative class of approaches applies graph-based semantic relatedness measures for extracting keywords [6] [12]. Some variants of these algorithms use the graph generated from a Wikipedia ontology [1] [6] [19]. However, these techniques operate at document level instead of identifying domain-specific keywords which is the focus of our work. Similar to these techniques, we propose to use graph-based semantic relatedness methods in combination with the Wikipedia category graph [23] for achieving better precision and recall.

A recently proposed word graph technique called ExpandRank [21] makes use of similar documents (called neighbourhood) for the extraction of keywords from a document. This technique requires an input parameter which is a small number of neighbouring documents. However, for finding similar documents the technique uses cosine similarity which is computationally very expensive and practically inapplicable for large datasets (also for our study). Furthermore, the exploitation of a neighbourhood of documents may result in topic drift resulting in the extraction of noisy terms for a document [11].

The Information Retrieval research community is increasingly making use of the information richness in open-domain knowledge sources for improving the effectiveness of Web search applications. The use of open-domain knowledge resources has been investigated for query intent identification, document analysis and understanding, and for query expansion [8] [9] [10]. To the best of our knowledge, this paper is the first attempt to address the problem of extracting domain-specific keywords from short-text (where standard NLP techniques may fail). Furthermore, we differ from previous approaches [1] [6] [19] in that we use the relationship between Wikipedia articles and Wikipedia categories for semantic relatedness, and we apply the infomap [17] algorithm for community detection.

3 Methodology

In this section we discuss the method we propose to extract domain-specific keywords from a set of Web pages (in our case the Web pages crawled from University Web sites). To this aim we first create an index of titles of the crawled Web pages in order to build the collection from where the keywords have to be extracted. Furthermore, we also index the titles of Wikipedia articles. Then, we apply an intersection between the two indexes (by only considering 2-5 grams). We then generate a subset of the intersection by applying a community detection algorithm. In the last phase, we also add to the selected 2-5 grams the significant 1 grams (i.e., single terms such as 'Science'). As an optional step we show how to reduce all domain-specific keywords (i.e., n-grams) to single terms; this may be useful for single terms tag cloud applications. In the following subsections the phases applied by the proposed extraction process are explained.

3.1 Web Pages and Wikipedia Indexes

The aim of the first phase of the proposed extraction process is to generate Web Pages and Wikipedia indexes. In Procedure 1 the meta-algorithm implementing this phase is shown. The inputs are the set of Web pages and the data-set of Wikipedia. The output consists of two indexes: the index of the titles of Web pages and the index of the titles of Wikipedia articles. In Step 1, empty indexes are initialized. Steps 2-8 are aimed at creating an index of the titles of Web pages after stopwords removal, followed by the generation of possible n-grams (up to 5 grams) for each title. While generating this index, we maintain the frequency count of each n-gram. We call this index $Index_{web}$. Similarly, Steps 9-15 create an index of titles of the Wikipedia articles after stopwords removal. We call this index $Index_{wiki}$.

3.2 Intersection between Indexes

The aim of the second phase of the proposed extraction process is to calculate the intersection of the indexes obtained from Procedure 1 followed by the removal of some non-topical noise. The meta-algorithm is shown in Procedure 2; the inputs to this module are $Index_{web}$, $Index_{wiki}$, and a map we generate to highlight associations between Wikipedia articles and categories[3] ($Wikipedia_{artMapCat}$). The output consists of a refined index called $Index_{inter}$. In Step 1, the intersection between the indexes $Index_{web}$ and $Index_{wiki}$ is found while preserving the frequency count which was observed during the generation of $Index_{web}$. This step helps to discover meaningful key phrases (i.e., known phrases over Wikipedia). In Step 2, key phrases containing numeric information (such as '2 May') are filtered out (they are not representative of a topic). In Step 3, we remove key phrases that appear under non-topical (i.e., not domain specific) categories of Wikipedia (such as 'people', 'country', 'sports'). This is based on the rationale that Wikipedia categories related to people are mainly about people and not about topics. We call the index produced by this module $Index_{inter}$.

Procedure 1 Index generation module: indexGeneration()

```
Require: Set of crawled Web Pages($WebPages$), data-set of Wikipedia
    Articles($Wikipedia_{Articles}$)
 1: create empty $Index_{web}$, $Index_{wiki}$
 2: for all $page$ in $WebPages$ do
 3:     $text$ ← ExtractTitleText($page$)
 4:     $text$ ← RemoveStopwords($text$)
 5:     for $i = 1$ to 5 do
 6:         $Index_{web}$.push(extract grams of length $i$ from $text$)
 7:     end for
 8: end for
 9: for all $article$ in $Wikipedia_{Articles}$ do
10:     $text$ ← ExtractTitleText($artilce$)
11:     $text$ ← RemoveStopwords($text$)
12:     for $i = 2$ to 5 do
13:         $Index_{wiki}$.push(extract grams of $i$ from $text$)
14:     end for
15: end for
16: return $Index_{web}$, $Index_{wiki}$
```

3.3 Domain restriction via community detection

Procedure 3 is the core of our approach, i.e., it discovers domain-specific key phrases by applying a community detection algorithm. The rationale behind the application of a community detection algorithm at this stage is that it contributes to select high quality domain-specific keywords by exploiting the semantic relatedness within communities. The inputs to this module are the Wikipedia category graph and $Index_{inter}$. The output consists of an index of

[3] In Wikipedia an article falls into one or more categories and the map returns Wikipedia categories corresponding to a Wikipedia article.

domain-specific key phrases and a list of top communities. In Step 1, the index of key phrases is initialized. In Step 2, an undirected graph of the Wikipedia categories is generated, where a node represents a Wikipedia category, the edge represents relationship between categories[4], and the weight on an edge is defined as the sum of the number of articles belonging to the first category and those belonging to the second category node (as shown in Steps 3-7). In Step 8, an undirected multi-level infomap algorithm [5] [17] is applied for community detection among Wikipedia categories. The application of the infomap algorithm yields an assignment of each category to exactly one community. Some communities may contain many categories while other communities may contain just one category. In Steps 9-10, the top-k communities are found on the basis of the number of unique articles that they contain. A community contains several Wikipedia categories and each category contains articles. To understand what we mean by unique articles, let us consider a community that contains 'k' categories (discovered by the infomap algorithm) and 't' total articles. The 't-x' articles that are not contained in any other community (i.e., only specific to the considered community) are called unique articles. As a simple example, an article on chemistry may be unique to the community that contains the categories related to chemical sciences and it would not be mentioned in communities such as the community that contains categories related to political sciences. This way, if a community contains several unique articles then it becomes a strong representative of their domain of interest. Similarly, the less the number of unique articles in the community the less its chance to be representative of the domain of interest (of the collection). As an extreme case, a community may contain zero or one unique article, which means it may be considered as an outlier (having little or no association with the considered domain). For example, a community containing only a unique article in the Wikipedia category '1979_births' implies to be a random outlier (if academic documents are considered). We consider articles contained in the top-k ranked communities as being relevant to the domain of interest. Therefore, we declare top-k communities as being the most representative of the domain. Based on this we further reduce $Index_{inter}$ and we call this subset $Index_{domainPhrases}$ (as shown in Step 11-18).

Procedure 2 First pass of identifying key phrases: indexesIntersection()

```
Require: Indexes of titles of Web pages ($Index_{web}$) & Wikipedia articles ($Index_{wiki}$),
    Wikipedia article to category map ($Wikipedia_{artMapCat}$)
1: $Index_{simpleInter} \leftarrow (Index_{web} \cap Index_{wiki})$
2: $Index_{noNumInter} \leftarrow Index_{simpleInter}$ - All key phrases with Numeric values
3: $Index_{inter} \leftarrow Index_{noNumInter}$ - All key phrases mentioned under non-topical
    category of Wikipedia (discovered using $Wikipedia_{artMapCat}$)
4: return $Index_{inter}$
```

[4] A category of Wikipedia may or may not have a super category i.e., a kind of hierarchical category structure but not strict [23] as there could be rare cycles in it.

[5] The predecessor to the algorithm was found to be the best-known algorithm for the community detection problem [5].

Procedure 3 Application of community detection for identifying key phrases: communityDetection()

```
Require: Wikipedia category graph ($Wikipedia_{catgraph}$), index after the application
    of first pass $Index_{inter}$
 1: create empty $Index_{domainPhrases}$
 2: $graph \leftarrow$ make Wikipedia categories as undirected graph using $Wikipedia_{catgraph}$
 3: for all $edge$ in $graph.edges$ do
 4:     $superCategory_{numArticles} \leftarrow (Index_{inter} \cap edge.superCategory.articles).length$
 5:     $subCategory_{numArticles} \leftarrow (Index_{inter} \cap edge.subCategory.articles).length$
 6:     $edge[weight] \leftarrow superCategory_{numArticles} + subCategory_{numArticles}$
 7: end for
 8: $List_{communities} \leftarrow$ infomap_undirected($graph$)
 9: $OrderedList_{communities} \leftarrow$ Descending order $List_{communities}$ by number of unique
    articles in community
10: $Top_{communities} \leftarrow$ topk($OrderedList_{communities}$, 10)
11: for all $comm$ in $Top_{communities}$ do
12:     for all $category$ in $comm.categories$ do
13:         $Articles_{inter} \leftarrow Index_{inter} \cap category.articles$
14:         for all $article$ in $Articles_{inter}$ do
15:             $Index_{domainPhrases}$.push($article.title$)
16:         end for
17:     end for
18: end for
19: return $Index_{domainPhrases}$, $Top_{communities}$
```

3.4 Incorporating Single Terms

The aim of this phase is to expand the $Index_{domainPhrases}$ by including single terms as shown in Procedure 4. The inputs to this module are $Index_{domainPhrases}$ and $Top_{communities}$. The output consists of the index of domain-specific key terms and phrases. In Step 1, an index of single terms is initialized. In Steps 2-12, each category in $Top_{communities}$ is visited and the category name which is composed of two or more words is considered as a candidate for extracting single terms (as shown in Steps 5-9), e.g., 'Cell_biology' is composed of two terms, 'cell' and 'biology'. To the aim of selecting meaningful single terms (in the academic sites application as we describe in Section 4), we have applied a simple rule in the considered context, a single key term (topical) usually ends with some postfix, such as 'logy', 'ics', 'ulus'[6]. In Step 13, we merge the extracted key terms with $Index_{domainPhrases}$ to produce the final output $Index_{domainTermsandPhrases}$.

3.5 Complete Algorithm

Procedure 5 shows the complete algorithm, the phases of which have been described in Sections 3.1–3.4. The inputs to this module are the set of Web pages and data-set of Wikipedia. The output consists of the index of domain-specific

[6] Similar type heuristic rules can be crafted or learned for other domains.

key terms and phrases. In Step 1, indexes for the Web pages and the Wikipedia data-set are generated by calling Procedure 1. In Step 2, the index is refined by creating the intersection of the indexes by making a call to Procedure 2. In Step 3, the index is further refined through community detection by using Procedure 3. Finally, the index is expanded to incorporate domain-specific key terms by making a call to Procedure 4.

Procedure 4 Final pass for identifying single key terms: expandtoSingleTerms()

```
Require: $Index_{domainPhrases}$, $Top_{communities}$
 1: create empty $Index_{SingleTerms}$
 2: for all $comm$ in $Top_{communities}$ do
 3:     for all $category$ in $comm.categories$ do
 4:         if $|category.\text{words}| > 1$ then
 5:             for all $word$ in $category.splitWords$ do
 6:                 if $word$.matchesRule() then
 7:                     $Index_{SingleTerms}$.push($word$)
 8:                 end if
 9:             end for
10:         end if
11:     end for
12: end for
13: $Index_{domainTermsandPhrases} \leftarrow Index_{domainPhrases}$.merge($Index_{SingleTerms}$)
14: return $Index_{domainTermsandPhrases}$
```

Procedure 5 Full algorithm: extractKeyterm_phrase()

```
Require: Set of crawled Web Pages($WebPages$), data-set of Wikipedia Articles
    ($Wikipedia_{Articles}$, Wikipedia category graph ($Wikipedia_{catgraph}$), Wikipedia ar-
    ticle to category mapping ($Wikipedia_{artMapCat}$))
 1: $Index_{web}$, $Index_{wiki} \leftarrow$ indexGeneration($WebPages, Wikipedia_{Articles}$)
 2: $Index_{inter} \leftarrow$ indexesIntersection($Wikipedia_{artMapCat}$, $Index_{web}$, $Index_{wiki}$)
 3: $Index_{domainPhrases}$, $Top_{communities} \leftarrow$ communityDetection($Wikipedia_{catgraph}$,
    $Index_{inter}$)
 4: $Index_{domainTermsandPhrases} \leftarrow$ expandtoSingleTerms($Index_{domainPhrases}$,
    $Top_{communities}$)
 5: return $Index_{domainTermsandPhrases}$
```

3.6 Application of the proposed algorithm to the Extraction of Single Key Terms

In this section, we show the application of the proposed algorithm for extracting important single terms instead of key phrases. This application can be useful for generating tag clouds of single key terms. To this aim, the index of domain-specific key terms and phrases is reduced to a list of single terms while preserving the frequency count of each term in the key term/phrase in such a way that none of the terms are over-counted. For example, consider there were only two n-grams in the index; 'a b' with frequency 'n' and 'b c' with frequency 'n'. Upon reducing to single terms we can say ‘a’ and ‘c’ occurs with ‘n’ frequency. However we can’t say with certainty that ‘b’ occurs with frequency ‘2n’ as it may not be correct in the case the n-gram is produced by a stream of data like ‘a b c’. Therefore in order to overcome the problem of over-counting, we maintained positional indexes of words within the stream (i.e., position of n-gram per Web page title). So now for the stream ‘a b c’ the positional index of ‘a’ is one, of ‘b’ is two and of ‘c’ is three, whereas for the discovered n-grams ‘a b’ and ‘b c’ there is a same positional index for both the ‘b’ terms therefore its frequency would be counted just once (this will aid in avoiding over-counting problem).

As a final step, we lemmatize all the obtained terms in order to use a conceptual representation of a term (e.g., sciences becomes science). Finally, all terms over all n-grams are arranged via frequency count, and the term having highest frequency represent the most important key term.

4 Experiments and Results

In this section we present the employed dataset, the evaluation measures, and the experimentations. We also present a discussion on the obtained results.

4.1 Dataset and Evaluation Measures

To evaluate the proposed approach we focus on academic Web sites for the identification of domain-specific key terms/phrases. To this aim, we crawled the English Web pages of eight post-graduate school Web sites from five different countries as shown in Table 1. For each Web site, we crawled up to the depth of five from the root page in order to cover at least 80%-95% of important Web pages [2]. In addition, to avoid a crawler trap, i.e. infinite dynamic Web pages such as calendars, we adopted the policy to crawl a maximum of the first 500 instances of each dynamic Web page.

To the aim of performing the evaluations we use the metric of Precision at k (P@k) results. P@k is defined by the ratio of correctly matching results over the first top-k results.

4.2 Evaluations

We conducted two experiments; in the first experiment we evaluated the quality of the methodology proposed in Section 3.5, and in the second experiment we evaluated the quality of the extension proposed in Section 3.6. For both experiments, 13 human annotators[7] made the relevance judgements for the top-20 results by associating a label *relevant*, *irrelevant* or *uncertain* with each keyword. For each keyword, the 13 judgements are aggregated to produce a single label: a keyword is labelled as *relevant* (or *irrelevant*) if the majority[8] of the annotators

[7] Except one, all the annotators have completed (at least) their post-graduate studies.
[8] In case of a tie we assign a random label (i.e., either *relevant* or *irrelevant*).

labelled it as *relevant* (or *irrelevant*). The keywords labelled as *uncertain* play no role in the aggregation process. The aggregated judgement is used in the graphical representations of the experiments.

Before conducting experiments, we produced four variants of the proposed methodology for the identification of domain-specific key terms/phrases, explained below:

n-grams: the basic algorithm (baseline) that uses $Index_{web}$ of Section 3.1 ordered by (descending) frequency of each n-gram.

simple_inter: the algorithm that uses $Index_{simpleInter}$ of Section 3.2 ordered by (descending) frequency of each n-gram.

intersect_noNum: the algorithm that uses $Index_{noNumInter}$ of Section 3.2 ordered by (descending) frequency of each n-gram.

complete: the algorithm that uses $Index_{domainTermsandPhrases}$ of Section 3.5 ordered by (descending) frequency of each n-gram.

Table 1. Dataset of school Web sites

School	**Convention**	**Website**	**Location**
IBA Karachi Campus	IBA-KHI	www.iba.edu.pk	PK
FAST NU Karachi Campus	FAST-NU-KHI	www.khi.nu.edu.pk	PK
LUMS	LUMS	www.lums.edu.pk	PK
MMU Cyberjaya Campus	MMU	www.mmu.edu.my	MY
Milano-Bicocca	Milano	www.unimib.it/go/page/English[a]	IT
NUI Galway Campus	NUIG	www.nuigalway.ie	IE
Cambridge	Cambridge	www.cam.ac.uk	UK
Oxford	Oxford	www.ox.ac.uk	UK

[a] The URL has now changed to www.unimib.it/go/102/Home/English

Experiment 1 In this experiment, we compared four variations of the proposed methodology to evaluate the capability to generate high quality domain-specific key terms/phrases. We asked annotators to label a key term/phrase as relevant when it correctly represents a complete name of a topical domain or sub-domain (academic topical area of interest). For instance,'Information Retrieval', 'Marine Biology', and 'Science' are relevant examples but 'Marine' is an irrelevant key term because it does not represent the name of a topical domain or sub-domain. In order to evaluate the agreement among annotators we calculated the value of Fleiss's Kappa [4], which showed high agreements (value 0.81).

Fig. 1(a) shows the quality of identifying domain-specific key terms/phrases at P@20. This figure shows that the ***complete*** algorithm outperforms the other algorithms. Fig. 1(b) shows the quality of noise elimination at steps ***simple_inter***, ***intersect_noNum***, and ***complete*** at P@20. Generally the graph shows very competitive precision in eliminating noisy information, however, the precision dropped for IBA-KHI in ***complete*** to 0.8 as some key terms/phrases were identified as noisy due to problems of disambiguation, e.g., 'computer studies' was disambiguated by the Wikipedia data set as 'computer' (instead of 'computer science'), which then led to non-topical categories (hence recognizing it as noisy information).

In our previous work [15], we conducted a similar experiment without incorporating single terms as explained in Section 3.4 in ***complete*** algorithm (as explained in Section 3.5) and found similar finding i.e., it outperformed the rest of algorithms, however incorporating single terms makes it even better.

Experiment 2 In this experiment, we evaluated the capability to generate high quality domain-specific key terms (i.e., single terms only). We asked annotators to label a key term as relevant when it correctly represents a complete or partial name of a topical domain or sub-domain (academic topical area of interest). For instance, 'Science' and 'Biology' are relevant examples, and so is 'Marine' if it represents a partial representation of 'Marine Biology'. As with the previous experiment, we calculated Fleiss's Kappa and found the value of 0.77, showing a high agreement among annotators.

In this experiment, we compared three well known algorithms i.e., BM25[16], TF-IDF and TF-Norm (term frequency normalized) against the three variations of our methodology ***n-grams***, ***simple_inter***, and ***complete***. The indexes generated by the three variations were processed to generate single terms respectively as discussed in Section 3.6. Fig. 2 shows that ***complete*** outperforms the other algorithms. However, there is still a room for improvement for the best case.

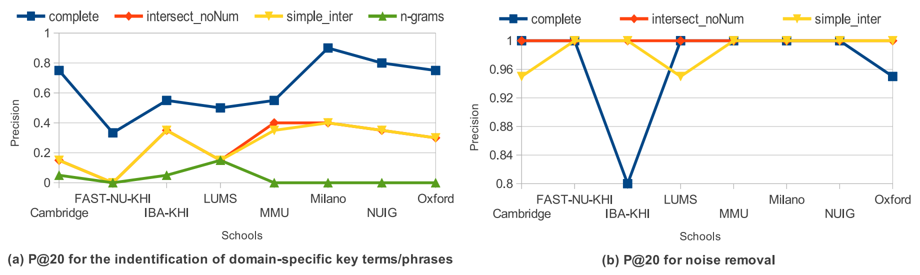

Fig. 1. P@20 results for experiment 1

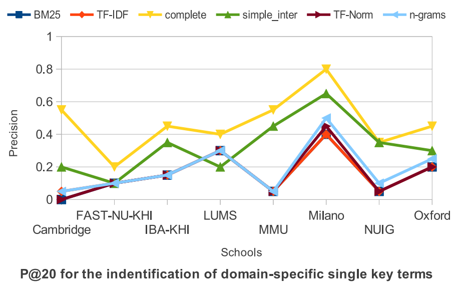

Fig. 2. P@20 results for experiment 2

To provide an illustration of typical results, Table 2 shows the data from the Milano-Bicocca Web site. In this table, we show top-15 domain-specific single key terms detected, domain-specific key terms/phrases detected, and noisy information that was eliminated by ***complete***.

Table 2. Data from Milano-Bicocca Web site

Type	**Extracted Data**
Single Key Terms Only	science, statistic, mathematics, computer, sociology, business, biotechnology, technology, developmental, psychology, service, material, social, political, communication
Key Terms/Phrases	science, economics, technology, psychology, mathematics, statistics, physics, sociology, medicine, law, biotechnology, developmental psychology, computer science, materials science, surgery
Noisy Information	research university, masters university, summer schools, doctoral degree, drop out, degree courses, campus university, student unions, ranking university, laboratory techniques, department statistics, union university, marco polo, department biotechnology, erasmus university

5 Conclusion and Future Work

In this contribution we presented an approach for identifying domain-specific key terms/phrases using Wikipedia. Furthermore, we have also presented an extension of our approach for identifying domain-specific single key terms only, which could be useful for some applications such as single word tag cloud definition. The evaluations have shown promising results in overall. In future, we would like to address the problem of disambiguation, and we would like to apply our methodology on the full text of Web pages.

References

1. *Wikipedia in Action: Ontological Knowledge in Text Categorization*, 2008.
2. R. Baeza-yates and C. Castillo. Crawling the infinite web: five levels are enough. In *In Proceedings of the third Workshop on Web Graphs (WAW*, pages 156–167. Springer, 2004.
3. D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, Mar. 2003.
4. J. Fleiss et al. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76(5):378–382, 1971.
5. S. Fortunato. Community detection in graphs. *Physics Reports*, 486(3-5):75 – 174, 2010.
6. M. Grineva, M. Grinev, and D. Lizorkin. Extracting key terms from noisy and multitheme documents. In *Proceedings of the 18th international conference on World wide web*, WWW '09, pages 661–670, New York, NY, USA, 2009. ACM.
7. K. S. Hasan and V. Ng. Conundrums in unsupervised keyphrase extraction: making sense of the state-of-the-art. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, COLING '10, pages 365–373, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

8. J. Hu, G. Wang, F. Lochovsky, J. tao Sun, and Z. Chen. Understanding user's query intent with wikipedia. In *WWW '09: Proceedings of the 18th international conference on World wide web*, pages 471–480, New York, NY, USA, 2009. ACM.
9. A. Jain and M. Pennacchiotti. Open entity extraction from web search query logs. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 510–518, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.
10. A. Kotov and C. Zhai. Tapping into knowledge base for concept feedback: leveraging conceptnet to improve search results for difficult queries. In *Proceedings of the fifth ACM international conference on Web search and data mining*, WSDM '12, pages 403–412, New York, NY, USA, 2012. ACM.
11. Z. Liu, X. Chen, Y. Zheng, and M. Sun. Automatic keyphrase extraction by bridging vocabulary gap. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, CoNLL '11, pages 135–144, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.
12. Z. Liu, P. Li, Y. Zheng, and M. Sun. Clustering to find exemplar terms for keyphrase extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1 - Volume 1*, EMNLP '09, pages 257–266, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.
13. R. Mihalcea and A. Csomai. Wikify!: linking documents to encyclopedic knowledge. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, CIKM '07, pages 233–242, New York, NY, USA, 2007. ACM.
14. D. Milne and I. H. Witten. Learning to link with wikipedia. In *Proceedings of the 17th ACM conference on Information and knowledge management*, CIKM '08, pages 509–518, New York, NY, USA, 2008. ACM.
15. M. A. Qureshi, C. O'Riordan, and G. Pasi. Short-text domain specific key terms/phrases extraction using an n-gram model with wikipedia. In *CIKM*, pages 2515–2518, 2012.
16. S. E. Robertson, S. Walker, M. Hancock-Beaulieu, and M. Gatfor. Okapi at trec-3. In *The Third Text REtrieval Conference TREC-3*, 1995.
17. M. Rosvall and C. T. Bergstrom. Multilevel compression of random walks on networks reveals hierarchical organization in large integrated systems. *PLoS ONE*, 6(4):e18209, 04 2011.
18. G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. In *INFORMATION PROCESSING AND MANAGEMENT*, pages 513–523, 1988.
19. P. P. Talukdar and F. Pereira. Experiments in graph-based semi-supervised learning methods for class-instance acquisition. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 1473–1481, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.
20. P. D. Turney. Learning algorithms for keyphrase extraction. *Inf. Retr.*, 2:303–336, May 2000.
21. X. Wan and J. Xiao. Single document keyphrase extraction using neighborhood knowledge. In *Proceedings of the 23rd national conference on Artificial intelligence - Volume 2*, AAAI'08, pages 855–860. AAAI Press, 2008.
22. I. H. Witten, G. W. Paynter, E. Frank, C. Gutwin, and C. G. Nevill-Manning. Kea: practical automatic keyphrase extraction. In *Proceedings of the fourth ACM conference on Digital libraries*, DL '99, pages 254–255, New York, NY, USA, 1999. ACM.
23. T. Zesch and I. Gurevych. Analysis of the Wikipedia Category Graph for NLP Applications. In *Proceedings of the TextGraphs-2 Workshop (NAACL-HLT)*, 2007.