Journal of Information Science
XX (X) pp. 1-16
© The Author(s) 2012
Reprints and Permissions:
sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/016555150000000
jis.sagepub.com
SAGE

Automatic Keyphrase Annotation of Scientific Documents Using Wikipedia and Genetic Algorithms

Arash Joorabchi
Department of Electronic and Computer Engineering, University of Limerick, Ireland

Abdulhussain E. Mahdi
Department of Electronic and Computer Engineering, University of Limerick, Ireland

Abstract
Topical annotation of documents with keyphrases is a proven method for revealing the subject of scientific and research documents to both human readers and information retrieval systems. This article describes a machine learning-based keyphrase annotation method for scientific documents which utilizes Wikipedia as a thesaurus for candidate selection from documents' content. We have devised a set of twenty statistical, positional, and semantical features for candidate phrases to capture and reflect various properties of those candidates which have the highest keyphraseness probability. We first introduce a simple unsupervised method for ranking and filtering the most probable keyphrases, and then evolve it into a novel supervised method using genetic algorithms. We have evaluated the performance of both methods on a third-party dataset of research papers. Reported experimental results show that the performance of our proposed methods, measured in terms of consistency with human annotators, is on a par with that achieved by humans and outperforms rival supervised and unsupervised methods.

Keywords
Text mining; scientific digital libraries; metadata generation; subject metadata; keyphrase annotation; keyphrase indexing; Wikipedia; genetic algorithms

1. Introduction

Scientific literature comprises scientific publications in form of journal articles, conference papers, technical reports, theses and dissertations, book chapters, and other materials about the theory, practice, and results of scientific inquiry. These materials are produced by individuals or groups in academic and industrial research-centric organizations such as universities, research and development companies, national research labs, centres, and institutes. A great volume of scientific literature is published in electronic form on the Internet and is available through institutional repositories, digital libraries, publishers' websites, authors' webpages, etc. The sheer volume of scientific literature available on the Internet makes finding the most relevant and up-to-date materials and information challenging. For example, reportedly the number of new papers published in the field of biomedical science alone exceeds 1,800 a day [1].

Annotating scientific documents with keyphrases as subject/topical metadata helps both humans and information retrieval systems to focus their search and discovery efforts on the most relevant items of interest and reduces the recall effort (i.e., ratio of desired to examined) [2, 3]. However, despite the fact that authors of scientific literature, especially those published in journals and conference proceedings, are encouraged and often required by editors to provide a list of keyphrases, scientific documents with manually assigned keyphrases by either authors or professional annotators are still in the minority. For example, as part of their work with the New Zealand Digital Library (www.nzdl.org), Steve

Corresponding author:
Abdulhussain E. Mahdi, Department of Electronic and Computer Engineering, University of Limerick, Limerick, Republic of Ireland.
Email: Hussain.Mahdi@ul.ie

Jones and Gordon Paynter [3] quantified the number of scientific documents with keyphrases in two sample collections. One collection contained the bibliographic details of more than 15,000 documents in the field of Human Computer Interaction (HCI) from which less than a third contained author-specified keyphrases. The other collection contained more than 40,000 computer science technical reports from which only 1,800 contained recognizable keyphrases. Our own examination of large-scale scientific digital libraries, such as CiteSeer[1] and arXiv[2], shows that this issue is even more dominant in the case of technical reports, theses, and dissertations which do not necessarily go through an editorial process before publication. To tackle this problem, researchers working in fields of Information Retrieval (IR) and Machine Learning (ML) have developed a wide range of automated keyphrase annotation methods. In general, these methods can be divided into two main categories:

1. Keyphrase extraction: keyphrases are picked from a set of candidate phrases extracted from the content of the document itself and are ranked and filtered based on their various statistical and/or semantical properties/features, such as frequency, position, length, and coherence, i.e., semantic relatedness to other terms in the document. The ranking function could be either (a) unsupervised, where it is heuristically defined by manual analysis of sample documents and encoding general properties of typical keyphrases, e.g., see [4, 5]; or (b) supervised, where it is automatically derived by a general-purpose ML algorithm from a training dataset, e.g., see [6-9]. Keyphrase extraction approach has two main weaknesses: 1) it is prone to generating phrases composed of a set or sequence of words that occur contiguously within the document and have statistically significant properties, such as high frequency (a.k.a statistically motivated phrases), but are ill-formed, grammatically wrong, or meaningless; 2) it limits the scope of potential candidates to the phrases appearing in the document, which in turn limits the highest achievable recall, regardless of the extraction algorithm deployed. For example, the first keyword of this article is “text mining” reflecting one of its broad topics/concepts, but it does not appear in the text explicitly.
2. Keyphrase assignment: keyphrases are picked from controlled vocabularies, such as taxonomies, thesauri, and subject heading systems (e.g., LCSH, MeSH, AGROVOC, Eurovoc) and are not confined to the phrases appearing in the document. In this approach, keyphrase annotation is treated as a multi-label text classification problem and general-purpose ML algorithms such as, Support Vector Machines (SVMs), Naïve Bayes (NB), and Decision Trees (DTs) are utilized to learn a model for each term in the controlled vocabulary from a set of manually annotated training documents. The learnt models are then applied to test documents for classification resulting in a set of high-probability classes (i.e., keyphrases) per document, e.g., see [10, 11]. Using this approach, assigned keyphrases are well formed, grammatically correct, and not limited to those appearing in the document. Therefore, it can cope with cases where a concept is discussed but not explicitly mentioned in the document (e.g., “text mining” in the case of the current article). However, depending on the characteristics of the target domain, this approach may suffer one or more drawbacks common among supervised ML-based approaches to IR in general, including lack of high quality and/or quantity training data, data sparsity and/or skewed distribution, and concept drift.

Medelyan and Witten [12, 13] proposed a hybrid approach as an intermediate between keyphrase extraction and keyphrase assignment which they have called keyphrase indexing. In this approach, candidate phrases are limited to a set of descriptors, i.e., preferred and commonly used terms for the represented concepts in a domain-specific thesaurus, which either themselves or their synonyms/alternative lexical forms (a.k.a non-descriptors, encoded in form of semantic relations in the thesaurus) occur in the document. This method of candidate generation eliminates the two above-mentioned weaknesses of keyphrase extraction approach as the generated candidate phrases are well-formed, semantically rich, and not restricted to those occurring in the document explicitly. Similar to keyphrase extraction, in this approach an unsupervised or supervised ranking function is deployed to model the general properties of keyphrases in order to rank and filter the most probable ones. This method of rank and filtering requires either no or limited training data, depending on the type of ranking function deployed. This is in contrast to keyphrase assignment approach which requires a set of annotated documents per descriptor. The main weakness of the keyphrase indexing approach is that it assumes there exists a comprehensive domain-specific thesaurus for the target domain, which is not always a feasible assumption. This weak point has been addressed by automatic construction of a universal thesaurus from Wikipedia [14, 15] and replacing the domain-specific thesauri with the thesaurus derived from Wikipedia [16, 17].

In this work, we aim to extend the keyphrase indexing approach, described above, by: (a) introducing a new set of features for the candidate phrases derived from Wikipedia, which enhances the performance of rank and filtering

[1] citeseerx.ist.psu.edu

[2] http://arxiv.org

process, and (b) introducing a new supervised ranking function based on Genetic Algorithms (GA) which eliminates the need for manual feature selection and outperforms general-purpose ML algorithms used for keyphrase annotation.

The rest of the article is organized as follows: Section 2 describes the process of generating candidate phrases and the candidate's features derived from Wikipedia. Section 3 first introduces a simple unsupervised ranking function, and then describes how it can be enhanced to a supervised ranking function based on genetic algorithms. Section 4 describes the evaluation process and presents its results. Section 5 discusses the evaluation results. This is followed by Section 6 which provides a conclusion along with a summary account of planned future work.

2. Candidate Generation

Following the work of Medelyan and Witten [16, 17], we utilize an open-source toolkit called Wikipedia-Miner [18] for candidate generation. Wikipedia-Miner effectively unlocks the Wikipedia as a general-purpose knowledge source for natural language processing (NLP) applications by providing rich semantic information on topics and their lexical representations beyond that offered by domain-specific thesauri. We use the topic detection functionality of the Wikipedia-Miner to extract all the Wikipedia topics (i.e., Wikipedia articles) whose descriptor or non-descriptor lexical representations occur in the document, and use the descriptors of the extracted topics as candidate phrases for the document. We have devised a set of twenty statistical, positional, and semantical features for candidate topics/phrases to capture and reflect various properties of those candidates which have the highest keyphraseness probability:

1. Term Frequency (TF): the occurrence frequency of the candidate phrase (i.e., descriptor of the extracted Wikipedia topic) and its synonyms and alternative lexical forms/near-synonyms (i.e., non-descriptors of the extracted Wikipedia topic) in the document. The TF values are normalized by dividing them by the highest TF value in the document. This results in real numbers in the range (0.0,1.0] and prevents high-frequency candidates in long documents from biasing the ranking function. The well-known TFIDF [19], which is a common term-weighting scheme in IR, has been used by many of the keyphrase annotation algorithms proposed in the literature. However, based on our own empirical experiments and those reported in [17, 20, 21], TF scheme outperforms the more sophisticated TFIDF in keyphrase annotation tasks. One reason for this could be the fact that a large training dataset is required to derive accurate IDF values for candidate phrases (usually not available), however Kim and Kan [22] show that even after adopting corpus-independent IDF values derived from Google N-gram corpus containing terabytes of data, the impact of TFIDF remains minimal.

2. First Occurrence: the distance between the start of the document and the first occurrence of the candidate topic, measured in terms of the number of characters and normalized by the length of the document. This feature reflects the observation that candidates occurring close to the beginning of documents, such as title, abstract, and introduction sections, have a higher keyphraseness probability [6, 7].

3. Last Occurrence: the distance between the end of the document and the last occurrence of the candidate topic, measured in terms of the number of characters and normalized by the length of the document. This feature reflects the observation that candidates occurring close to the end of documents, such as discussion, conclusion, and references sections, have a higher keyphraseness probability [5, 6, 17].

4. Occurrence Spread: the distance between the first and last occurrences of the candidate topic, measured in terms of the number of characters and normalized by the length of the document. This feature reflects the observation that candidates which are more evenly spread within the document have a higher keyphraseness probability. For example, the candidate phrase "keyphrase assignment" only appears in the introduction section of this article, where we discuss the related work, whereas, the candidate phrase "keyphrase annotation", which is one of the keyphrases assigned to the article, is spread throughout the article. A feature comparison study by Medelyan [17] shows that occurrence spread generally outperforms both the first and last occurrence features in terms of the Area Under the ROC Curve (AUC).

5. Length: the number of words in the candidate phrase, i.e., the descriptor of the candidate topic. This feature reflects the general observation that multi-word phrases have a higher keyphraseness probability as they tend to be more specific and less ambiguous. The keyphrase annotation studies which adopt this feature (e.g., see [7, 13, 16, 17, 23]) compute the length of a candidate phrase by simply counting its number of words or characters. However, our approach is to: (a) split the hyphenated words, (b) count the stopwords as 0.5 and non-stopwords as 1.0, (c) normalize the count value by dividing it by 10.0, (d) eliminate candidates which either have a normalized value greater than 1.0 or those which do not contain any letters (e.g., numbers, numerical dates). Using this weighting scheme reduces some of the noise introduced to the length feature by stopwords. For example, the phrase "de-hyphenation" would count as 1.5 words since "de" is a stopword, and its normalized length value would be 0.15. Eliminating candidates with normalized length values of greater than 1.0 restricts valid candidates to those containing a maximum of 10 non-stopwords or combinations of stop and non-stop words with a length (measured using the weighting scheme described above) not exceeding 10.0. This is a rather high value for maximum length compared to that adopted by previous works, which usually do not include candidate phrases longer than 3-5 words (counting stopwords as equal as non-stopwords). However, our analysis of a recent dump of the English Wikipedia from July 2011 [24], shows that for a total of 3,573,789 topics, 522,512 (14.6%) have a length (measured using our weighting scheme) in range of 3.5-5.0, 155,220 (4.3%) have a length in range of 5.5-10.0, and 4,083 (0.1%) have a length in range of 10.5 up to a maximum of 32.0. Based on this observation we decided to include all the candidates with a length value up to 10.0 and, hence, excluded only 0.1% of potential candidates, which are highly unlikely to be picked by human annotators.

6. Lexical Diversity: the descriptor and non-descriptors of a given topic could appear in a document in various lexical forms. We calculate the lexical diversity by (a) case-folding and stemming all the lexical forms of the candidate topic which appear in the document, using an improved version of Porter stemmer [25]called the English (Porter2) stemming algorithm [26]; (b) counting the number of unique stems minus one, so that the lexical diversity value would be zero if there is only one unique stem. Lexical diversity values are normalized by dividing them by the highest possible lexical diversity value between all topics in Wikipedia. In the Wikipedia dump used in this work, the topic “Roman numerals” happens to have the highest lexical diversity value of 2,075. As explained in Section 3, this feature is only used in the supervised ranking function to balance and complement the lexical unity feature.

7. Lexical Unity: inverse of lexical diversity calculated as: $1.0 - \textit{lexical diversity}$. Our assumption is that the candidates with higher lexical unity values would have a higher keyphraseness probability.

8. Average Link Probability: the average value of the link probabilities of all the candidate topic’s lexical forms which appear in the document. The link probability of a lexical form is the ratio of the number of times it occurs in Wikipedia articles as a hyperlink (directing to its corresponding article) to the number of times it occurs as plain text.

9. Max Link Probability: the maximum value of all link probabilities of the lexical forms for a candidate topic which appear in the document. Both the average and max link probability features are based on the assumption that candidate topics whose descriptor and/or non-descriptor lexical forms appearing in the document have a high probability of being used as a hyperlink in Wikipedia articles, also would have a high keyphraseness probability.

10. Average Disambiguation Confidence: in many cases a term from the document corresponds to multiple topics in Wikipedia and hence needs to be disambiguated. For example, the term “Java” could refer to various topics, such as “Java programming language”, “Java Island”, “Java coffee”, etc. As described in [27], the Wikipedia-Miner uses a novel ML-based approach for word-sense disambiguation which yields an F-measure of 97%. We have set the disambiguator to perform a strict disambiguation, i.e., each term in the document can only correspond to a single topic which has the highest probabilistic confidence. The value of the *average disambiguation confidence* feature for a candidate topic is calculated by averaging the disambiguation confidence values of its descriptor and non-descriptor lexical forms that appear in the document.

11. Max Disambiguation Confidence: the maximum disambiguation confidence value among the lexical forms of a candidate topic which appear in the document. Both the average and max disambiguation confidence features are incorporated into the ranking function to reduce the likelihood of candidate topics with low disambiguation confidence values being ranked as top keyphrases. This is because low disambiguation confidence values for a candidate topic shed doubt on its existence and validity in the document.

12. Link-Based Relatedness to Other Topics: the Wikipedia-Miner measures the semantic relatedness between topics using a new approach called Wikipedia Link-based Measure (WLM). In this approach the relatedness between two Wikipedia articles/topics is measured according to the number of Wikipedia topics which discuss/mention and have hyperlinks to both the two topics being compared (see [28] for details). For example, the two key phrases/topics assigned to this article, “text mining” and “genetic algorithms” have 53% relatedness based on the fact that a third Wikipedia topic “artificial intelligence” has mentioned and have hyperlinks to both. The *link-based relatedness to other topics* feature value of a candidate is calculated by measuring and averaging its relatedness to all the other candidates in the document.

13. Link-Based Relatedness to Context: the only difference between this feature and the *link-based relatedness to other topics* is that the relatedness of the candidate topic is only measured against those of other candidate topics in the document which are unambiguous, i.e., their descriptor and non-descriptor lexical forms occurring in the document have only one valid sense. Both the *link-based relatedness to context* and *link-based relatedness to other topics* features are designed to increase the likelihood of those candidate topics with high semantic relevance to other topics in the document being picked as top keyphrases. However, the former only takes into account the unambiguous topics in the document and therefore has high accuracy but low coverage, whereas the latter also includes the ambiguous topics which have been disambiguated based on their surrounding unambiguous context (i.e., unambiguous topics in the document) and therefore has lower accuracy but conclusive coverage.

14. Category-Based Relatedness to Other Topics: since May 2004, Wikipedia authors have been categorizing Wikipedia articles according to a community-built classification scheme (a.k.a folksonomy) which has been growing rapidly. The English Wikipedia dump from July 2011, which has been used in this work, contains a total of 739,980 unique categories. This shows 809% growth since January 2006 when it was reported to contain only 91,205 categories [29]. However, in contrast to traditionally expert-built classification schemes and taxonomies, such as Dewey Decimal Classification (DDC) and Library of Congress Classification (LCC) which adhere to a hierarchical tree structure, the Wikipedia classification scheme has a loose semi-hierarchical directed-graph structure which allows articles to belong to multiple categories and categories to have multiple parents. The collaborative and crowdsourcing nature of taxonomy development and categorization work in Wikipedia makes it prone to some level of noise. For example, our analysis of the Wikipedia dump used in this study and those done by others (e.g., see [30]) have shown the existence of self-loops ($C_1 \rightarrow C_1$), direct-loops ($C_1 \rightarrow C_2 \rightarrow C_1$), and indirect-loops (e.g., $C_1 \rightarrow C_2 \rightarrow C_3 \rightarrow C_1$) among some categories in Wikipedia classification graph. Our study shows that as of July 2011, 95% of Wikipedia articles are classified and on average each classified article belongs to 3.82 categories. When a candidate topic is classified, we can utilize its categorization data to measure its semantic relatedness to other candidates in the document. One of the well-known approaches to estimate the conceptual relatedness between two topics in a taxonomy is to measure the distance of the shortest path between the two nodes in terms of the number of edges along the path, first proposed by Rada et al. [31] in 1989. An enhanced version of this approach, which counts the number of nodes instead of edges along the shortest path and normalizes the resulting distance by dividing it by two times the maximum depth of the taxonomy (as the longest possible distance), was proposed in 1998 by Leacock and Chodorow [32], and used to measure the relatedness between two terms in WordNet as:

$$\text{Relatedness}(term_1, term_2) = -\log \frac{\text{Distance}(term_1, term_2)}{2 \times Maximum\ depth\ of\ the\ taxonomy} \qquad (1)$$

In 2006, Strube and Ponzetto [29] adopted this measure to estimate the semantic relatedness between two topics in Wikipedia and showed its superiority compared to other measures proposed in the literature up to then. In 2008, Milne and Witten [28] showed that their WLM approach, implemented in Wikipedia-Miner and utilized in this work (features 12 and 13), outperforms the shortest-path approach. Nevertheless, we believe deploying these two approaches together would improve the overall performance of keyphrase indexing, as they estimate the semantic relatedness of topics very differently using two independent information sources in Wikipedia and therefore could complement each other. We measure the category-based relatedness of two Wikipedia topics as:

$$\text{Relatedness}(topic_1, topic_2) = 1 - \frac{\text{Distance}(topic_1, topic_2) - 1}{2D - 3} \qquad (2)$$

where D is the maximum depth of the taxonomy, i.e., 16 in case of the Wikipedia dump used in this work. The distance function returns the length of the shortest path between $topic_1$ and $topic_2$ in terms of the number of nodes along the path. The term $2D-3$ gives the longest possible path distance between two topics in the taxonomy, which is used as the normalization factor, i.e., $2 \times 16 - 3 = 29$. The shortest possible distance between two nodes/topics is 1 (in case of siblings) and the longest is $2D - 3$. Therefore subtracting one from the outcome of the distance function results in a highest possible relatedness value of 1.0, e.g., $1 - (1 - 1) / (2 \times 16 - 3) = 1.0$, and a lowest possible relatedness value of 0.03, e.g., $1 - (29 - 1) / (2 \times 16 - 3) = 0.03$. Changing the divisor from $2D - 3$ to $2D - 4$ reduces the lowest possible relatedness value to zero, however we have adopted the former and instead assign a zero value to relatedness when either $topic_1$ or $topic_2$ are amongst the 5% of Wikipedia topics which are not classified. It should be noted that the relatedness function in Equation 1 is logarithmically decreasing whereas our proposed function in Equation 2 is linear. This is due to: (a) using logarithmic inversed distance for measuring relatedness over linear or exponential inversed distances has not been justified in [29, 32], (b) our supervised ranking function described in Section 3 scales each of the features either logarithmic, linear, or exponential according to the resulting effect on the performance of the model being learnt from training data. We have used an open-source toolkit for graph modelling, analysis, and visualization called JUNG [33], to build the classification graphs of the documents and measure the shortest path distance between the candidate topics. The value for *category-based relatedness to other topics* for each candidate is calculated by measuring and averaging its category-based relatedness to all the other candidates in the document.

As an example, figure 1 shows the classification graph of a sample document from the evaluation dataset used in Section 4 (Doc. ID: 287), visualized using Gephi [34]. This 10-page long paper titled “Clustering Full Text Documents” contains a total of 150 candidate topics which all, except one, are classified. Its classification graph consists of a total of 5,736 vertices (150 topics, 5,586 categories) of which 61 are root vertices, and a total of 16,295 edges. The resulted graph has the following properties: diameter of 14, density of 0.001, average degree of 5.68, average path length of 5.75, 32,884,490 shortest paths, and average clustering coefficient of 0.1. The topics/keyphrases “Statistics” which has the highest *category-based relatedness to other topics* feature value, 0.89, and “cluster analysis” which has been assigned to the document by 10 out of 15 teams of human annotators and has a relatedness value of 0.87 are magnified in the figure showing their immediate parent categories.

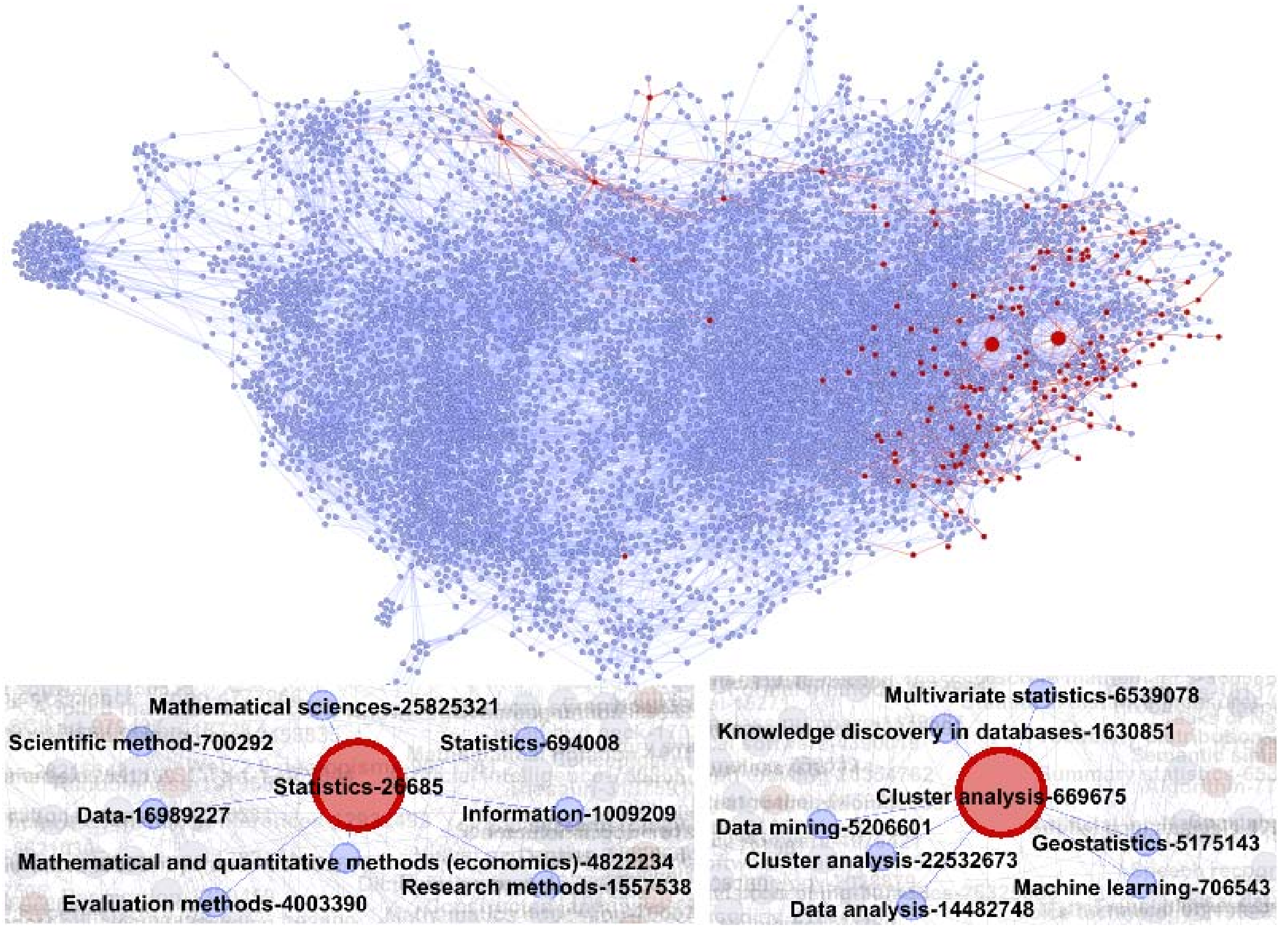

Fig. 1. Visualized classification graph of a sample document.

15. Generality: the depth of the topic in the taxonomy measured as its distance from the root category in Wikipedia, normalized by dividing it by the maximum possible depth, and inversed by deducting the normalized value from 1.0. It ranges between 0.0 for the topics farthest from the root and unclassified ones, and 1.0 for the root itself. Medelyan [17] has shown that there is a small tendency among human annotators to pick topics/keyphrases which are more general and belong to lower depths of the Wikipedia taxonomy, This is reflected by the generality feature.

16. Speciality: inverse of generality calculated as: $1.0 - \mathit{generality}$. This feature is only used in the supervised ranking function (see Section 3) to balance and complement the generality feature. In some domains or datasets more general keyphrases are preferred by human annotators, whereas in others more specific keyphrases are preferred. The supervised ranking function learns these preferences from the training data and reflects them in the ranking function by adjusting the impact of generality and speciality features simultaneously.

17. Distinct Links Count: total number of distinct Wikipedia topics which are linked in/out to/from the candidate topic, normalized by dividing it by the maximum possible distinct links count value in Wikipedia, which is equal to 307,035 and belongs to the topic “United States”. This feature reflects the assumption that candidate topics which are linked to by a larger number of unique topics in Wikipedia would have a higher keyphraseness probability.

18. Links Out Ratio: total number of distinct Wikipedia topics which are linked out from the candidate topic, divided by the *distinct links count* value of the candidate. Our preliminary experiments show that the candidates with a higher ratio of links out to links in, have a higher keyphraseness probability.
19. Links In Ratio: total number of distinct Wikipedia topics which are linked in to the candidate topic divided by the *distinct links count* value of the candidate. This feature is only used in the supervised ranking function (see Section 3) to balance and complement the *links out ratio*.
20. Translations Count: number of languages that the candidate topic is translated to in the Wikipedia, normalized by dividing it by the maximum possible translations count value in Wikipedia, which is equal to 216 and belongs to the topic "English language". This feature reflects the assumption that candidate topics which have been translated to more languages in Wikipedia would have a higher significance and keyphraseness probability.

3. Rank and Filtering

The number of candidate Wikipedia topics in a document could range from tens to thousands depending on the length of document. For example, the number of candidates in a collection of 20 research papers used in Section 4 as the evaluation dataset ranges from 131 for a 10-page paper to 708 for a 38-page paper with an average of 275 candidates per document. As discussed in Section 1, the function applied to rank all the candidates and filter out those with highest keyphraseness probabilities could be either supervised or unsupervised. Since all the features defined in Section 2 are normalized to range from 0.0 to 1.0, a simple unsupervised function could be defined as:

$$\text{Score}(topic_j) = \sum_{i=1}^{|F|} f_{ij} \tag{3}$$

which computes the sum of all feature values of a given candidate topic, *topic*$_j$, as its keyphraseness score. The feature set, *F*, does not contain the inverse features f_6, f_{16}, and f_{19} as they are only designed to be used in the supervised function. The main advantage of this unsupervised approach is that it does not involve a training process and, therefore, does not require any manually annotated documents for learning a rank and filtering function from. Hence, it may be readily applied to document collections across all domains with minimum effort. However, this approach forces a number of naive assumptions on the general properties of keyphrases in research documents, which negatively impact the accuracy performance of the rank and filtering function:

- All the summed features carry the same weight in respect to their capacity for measuring the keyphraseness probability of a candidate. This is a virtually impossible assumption as shown previously (e.g., see [17]).
- All the features correspond and contribute to the keyphraseness probability of candidates linearly. This assumption does not hold neither intuitively nor empirically [17]. For example, in case of positional features such as first occurrence (f_2) and last occurrence (f_3), only extreme values (>~0.9) should have a significant effect on the overall keyphraseness scores of candidates. Therefore, an exponential correspondence between the values of these features and the scores of candidates better captures the behaviour of these features. Another example is the term frequency (f_1) as it has been proven that logarithmically scaled frequency values better reflect the true significance of terms appearing in a document compared to raw frequency values [35]. This is due to the word burstiness phenomenon, i.e., the probability of a given term reappearing in a document grows exponentially as its frequency increases [36].

The above issues may be addressed to a large degree by adding a weight, w_i, and a degree parameter, d_i, to each feature in equation 3:

$$\text{Score}(topic_j) = \sum_{i=1}^{|F|} w_i \, f_{ij}^{\;d_i} \tag{4}$$

In an unsupervised setting, knowledge engineers would require to heuristically find and assign the (near) optimum values to these two parameters via examination of a collection of manually annotated documents. However, the optimum values for these parameters could change from one domain or dataset to another. In fact, it could even change from an individual or a group of annotators to another. For example, some human annotators have a high tendency to pick the keyphrases from the beginning sections of documents, and therefore modelling this behaviour requires a significant weight to be assigned to first occurrence (f_2). Another example of this is the case of generality (f_{15}), where in one domain or dataset more general keyphrases might be preferred, while in others more specific keyphrases would be preferred. Accordingly, the degree parameter of this feature needs to be adjusted such that it would be scaled exponentially in the former case and logarithmically (or set to zero to neutralize) in the latter. Since the optimum values for the weight and degree parameters highly depend on the nature of the dataset (e.g., its domain, genre, human annotators' preferences), there are no universal optimum values for these parameters. Hence, in order to achieve the best performance, the learning process needs to be repeated whenever the underlying nature of the dataset changes. In our supervised approach we automate this learning process by utilizing genetic algorithms to learn the optimum values for the weight and degree parameters from a set of manually annotated documents. In this approach, the learning process consists of the following steps:

1. Generating an initial population consisting of a set of ranking functions with random weight and degree parameter values from within a predefined range.
2. The fitness of each individual ranking function in the population is evaluated via applying it to a set of training documents to rank and filter their most probable keyphrases, and comparing the resulted top *n* keyphrases per document with those assigned by human annotators.
3. Based on their fitness, a number of individuals in the current population are stochastically selected, crossed, and mutated to form a new population for the next generation.
4. Steps 2 and 3 are repeated successively until one of the following termination conditions is reached:
    a. A ranking function with the highest possible fitness is found, i.e., for all the documents in the training set, all the top *n* keyphrases resulted from applying the ranking function match those assigned by human annotators. In practice we use the inter-indexer consistency measure to evaluate the fitness of individual ranking functions (see Section 4 for details of the evaluation measure used).
    b. The threshold on the number of generations is invoked. We have defined a greedy thresholding scheme which initially allows a predefined number of generations specified by the *threshold* variable to pass, and from that point on it counts the number of generations that pass without any improvement in the fitness of their fittest individual compared to that of the previous generation. Each time there is an improvement, the counter is reset to zero. The iteration process terminates when the number of generations passed without improvement equals half the total number of generations passed, i.e.,

    ```
    if (#generations_passed ≤ threshold) continue;
    else{
       if (no_improvement) counter++; else counter=0;
       if (counter ≥ #generations_passed/2) terminate;
    }
    ```

    It should be noted that since we use elitism the fitness of the best individual in each generation is guaranteed to be equal or higher than the fitness of the best individual in the previous generation, and the proposed thresholding mechanism would not work effectively without elitism. Also in practice, virtually always the threshold of the second condition is invoked to prevent over fitting before the first condition may be met, if at all possible.
5. The best individuals of all the generations built since the last generation with an improvement up to the last one before termination, are stored to be used for rank and filtering high probability keyphrases in unseen documents.

The degree parameters are float numbers with values between 0.0 and 2.0, allowing each feature, f_i, to be scaled logarithmically ($0.0<d_i<1.0$), linearly ($d_i=1.0$), exponentially ($1.0<d_i\leq 2.0$), or to become neutralized ($d_i=0.0$). The weight parameters have the same type and range as the degree parameters, allowing the weight of each feature, i.e., the magnitude of its impact on the total keyphraseness score of a given candidate, to become nil ($w_i=0.0$), a fraction of neutral ($0.0<w_i<1.0$), neutral ($w_i=1.0$), or up to twice bigger than neutral ($1.0<w_i\leq 2.0$). The defined ranges allow the genetic algorithm to render the features that are too noisy or counterproductive redundant via setting their degree to zero, or setting their weight to zero (or close to it). This in effect automates the feature selection process. For example, the GA is able to eliminate either the *generality* or *speciality* features or strike a balance between the two by adjusting their weight and degree parameters.

In the unsupervised setting, the universal ranking function defined in Equation 3 is applied to unseen documents and the top *n* keyphrases with the highest keyphraseness probability are filtered out. In the supervised setting however, the data stored at the fifth step of the learning process is used to apply an individual or a set of ranking functions defined in

Equation 4, whose weight and degree parameters are adjusted according to the general properties of keyphrases in the target dataset. We have developed and evaluated two different methods to this:

- **Last best:** in this method, simply the individual function with the highest fitness from the last generation before termination is applied to the documents to rank and filter their top *n* keyphrases.
- **Unique bests:** in this method, the final score of each candidate keyphrase in a given document is calculated as the sum of its scores from all the unique individual ranking functions created during the learning process, which have yielded the best fitness value (achieved before termination) with different weight and degree value sets. This ensembled scoring method takes into account all the variations of the weight and degree value sets which have yielded the final best fitness value, and therefore, it is expected to outperform the *last best* method.

In this work we have used an open-source implementation of genetic algorithms called ECJ [37] which is mature, well-documented, and can be easily customized. For example, the weight and degree parameters of each feature constitute a logical unit called a “chunk” which should not be broken during crossover, and this is supported in ECJ by allowing user-specified chunk boundaries. In the supervised setting described above, each individual genome in a generation consists of 40 genes of type float which are grouped in 20 weight-degree chunks for the 20 features described in Section 2. Details of the other parameter settings of the supervised learning and the GA, such as threshold value and population size, which offer a trade-off between model accuracy and training time depending on the user requirements, are discussed in Section 4.

4. Experimental Results & Evaluation

For the purpose of evaluating the performance of the proposed keyphrase annotation method, we have used a dataset called wiki-20 [38] created by Medelyan and Witten [16, 17]. The wiki-20 collection consists of 20 Computer Science (CS) related technical research reports, each manually annotated by fifteen different human teams independently. Each team consisted of two senior undergraduate and/or graduate CS students. The teams were instructed to assign about five keyphrases to each document from a controlled vocabulary of over two million terms which served as article titles (i.e. topic descriptors) in Wikipedia at the time the dataset was compiled. The teams have assigned an average of 5.7 keyphrases to each document and each document has received an average of 35.5 unique keyphrases. We follow the evaluation approach from [16, 17] and use the inter-indexer consistency formula proposed by Rolling [39] to measure the quality of keyphrases assigned to the test documents by our method via comparing them with those assigned by each team of human annotators:

$$\text{Inter-indexer consistency}(A,B) = \frac{2c}{a+b} \tag{5}$$

where *A* and *B* represent the two annotators (i.e., human and machine, human and human, machine and machine) whose inter-consistency is being measured, *a* and *b* are the number of terms assigned by each annotator, and *c* is the number of terms they have in common. The overall inter-indexer consistency score of an annotator is calculated by first measuring and averaging its inter-consistency with all the other annotators per test document, and then averaging the results over all the documents. This measure is also used in the second step of the learning process described in Section 3, to evaluate the fitness of individual ranking functions in a given population.

As shown in [13], Rolling’s inter-indexer consistency formula is equivalent to the well-known F1 measure, which is a widely used metric in information retrieval [40]. However, as discussed in [41], having only a single set of keyphrases assigned by a human annotator (individual or collaborating team) per document, taking it as the gold standard, and using the popular measures of precession, recall, and their harmonic mean, F1, to evaluate the quality of keyphrases assigned by the machine annotator, ignores the highly subjective nature of keyphrase annotation tasks [42]. For example, consider the outer circle illustrated in Figure 2 as the boundary for the set of valid keyphrases, *VK*, for a given document, HA_1, HA_2, and HA_3 as sets of keyphrases assigned by three different individuals or groups of human annotators, and *MA* as the set of keyphrases assigned by a machine annotator, i.e., automatic keyphrase annotation method being evaluated.

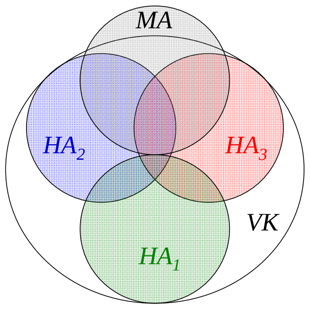

Fig. 2. Comparison of using single versus multiple gold standards for measuring the performance of a machine annotator.

In this example, if HA_1 was the only available set of human generated keyphrases to be used as the gold standard, then both the F1 and inter-indexer consistency scores of the machine annotator would be zero. This is because despite the large overlap between *MA* and *VK*, there is not overlap between *MA* and HA_1. Whereas when additional gold standards, HA_2 and HA_3, are available and used for evaluating the overall inter-indexer consistency score of the machine annotator, a more accurate estimation of the machine annotator's performance could be achieved. In fact, as can be seen in the illustration, if the quality of *MA* and HA_1 were to be compared with each other by using HA_2 and HA_3 as gold standards, the overall quality of *MA* would be significantly higher than HA_1, as $|MA \cap (HA_2 \cap HA_3)| > |HA_1 \cap (HA_2 \cap HA_3)|$. On the other hand, datasets of research documents with multiple sets of keyphrases assigned by different human annotators independently are rare and expensive to create in comparison to those with single set of keyphrases per document. The latter can be created automatically by mining the content of papers and articles archived in open-access scientific digital libraries, such as CiteSeer [43], and extracting author assigned keyphrases where available. Whereas, in case of the former, additional sets of keyphrases are usually not available and need to be created manually, for example the small wiki-20 dataset used in this work has taken ninety man-hours to create.

In order to achieve a thorough evaluation of the performance of our method, we have conducted three rounds of evaluation, each consisting of three sets of experiments: (ExpA) 2-fold cross-validation, (ExpB) 4-fold cross-validation, and (ExpC) 20-fold cross-validation, which corresponds to Leave-One-Out Cross-Validation (LOOCV). The data and results of all the experiments including the classification graph of all the documents in GraphML format, the test and training data used in each experiment, the learnt models in XML format and their visualization in gnuplot, and the resulted ranked keyphrases per document per experiment are available for download[3].

Table 1 presents the results of the three rounds of evaluation. The third and fourth columns list the average number of generations passed before the learning process terminates and the average training time per experiment measured in minutes. The fifth column lists the average highest fitness achieved on the training sets per experiment. The sixth column lists the average highest possible overall consistency that could be achieved with the human annotators on the training sets per experiment using an ideal rank and filtering function. The seventh and eighth columns list the results of applying the learnt ranking functions to the test sets using the *unique bests* method described in Section 3, and the average number of unique bests applied, respectively. The ninth column lists the results of applying the *last best* method per experiment. The tenth column lists the results of applying the universal unsupervised function to the test sets. The eleventh column lists the average highest possible overall consistency that could be achieved with the human annotators on the test sets per experiment using an ideal rank and filtering function.

As described in Section 3, the unsupervised rank and filtering function may be applied to a target dataset directly and only requires the user to specify the number of top high-probability keyphrases, *nk*, which should be assigned per document. However, the supervised function and its GA-based learning component have a number of extra parameters that need to be set before the learning process starts. These parameters and their corresponding values are listed in Table 2. It also should be noted that in all the experiments reported here, Version 1.20, Revision 233 of the Wikipedia-miner has been used for candidate generation. All the experiments have been conducted on a Dell Optiplex 780 mini-tower platform running Ubuntu 11.10 (64-bit) with an Intel Core 2 Duo Processor E8600 (6M Cache, 3.33 GHz, 1333 MHz FSB) and 4GB of DDR3 SDRAM.

In the first round of evaluation, we set the threshold to 400 (population-size multiplied by 10) which forces the GA to go through a minimum of 800 generations before terminating the learning process. In the second round, we reduced the threshold value from 400 to 200 to speed up the learning process and measure the effect it has on the quality of

[3] http://www.skynet.ie/~arash/zip/KA_Wiki20_WM1.2-R233_ECJ20.zip

results. In the third round however, we doubled the threshold from 400 to 800 to examine if having a larger threshold and lengthier learning process would result in an improved overall performance. The overall results of these three rounds, as shown in Table 1, indicate underfitting in case of the second round and overfitting in case of the third round, both resulting in an underperforming model.

Table 1. Results of the three rounds of evaluation.

Round	Dataset	Train				Test					Settings	
		Avg. # of Gens	Avg. Time Mins.	Avg. Fitness	Avg. Highest Possible	Supervised			Avg. Un-supervised	Avg. Highest Possible	Thre shold	*NK*
						Avg. Unique Bests	Avg. # of Unique Bests	Avg. Last Best				
First	ExpA (2-fold CV)	800	20.93	37.1%	48.5%	33.4%	202	32.7%	30.7%	48.5%	400	5
	ExpB (4-fold CV)	3137	131.03	37.4%	48.5%	32.7%	100	32.8%	30.7%	48.5%	400	5
	ExpC (LOOCV)	**1510**	**75.19**	**36.7%**	**48.5%**	**33.5%**	**73**	**33.5%**	**30.7%**	**48.5%**	**400**	**5**
Second	ExpA (2-fold CV)	672	17.07	37.6%	48.5%	32.8%	84	33.5%	30.7%	48.5%	200	5
	ExpB (4-fold CV)	584	24.34	36.8%	48.5%	31.4%	54	31.5%	30.7%	48.5%	200	5
	ExpC (LOOCV)	**1000**	**49.83**	**36.5%**	**48.5%**	**32.9%**	**53**	**32.6%**	**30.7%**	**48.5%**	**200**	**5**
Third	ExpA (2-fold CV)	2420	62.22	38.3%	48.5%	33.3%	194	33.5%	30.7%	48.5%	800	5
	ExpB (4-fold CV)	2487	102.55	37.5%	48.5%	32.9%	218	33.1%	30.7%	48.5%	800	5
	ExpC (LOOCV)	**3104**	**154.38**	**37.1%**	**48.5%**	**32.8%**	**149**	**31.8%**	**30.7%**	**48.5%**	**800**	**5**

Table 2. Genetic algorithm settings.

Species	Population Size	Genome Size	Chunk Size	Min Gene	Max Gene	Elites	Crossover Type	Selection Method	Mutation Type	Mutation Probability	Threads
Float	40	40	2	0.0	2.0	1	two points	Tournament	Reset	0.05	2

Table 3 compares the performance of our machine annotator on the wiki-20 dataset with human annotators, a baseline machine annotator based on TFIDF, two un-supervised machine annotators: the work of Grineva et al. [4], and CKE [5], and two supervised machine annotators: KEA++ (KEA-5.0) [13, 17] and Maui [17]. The supervised and unsupervised machine annotators with the highest performance results appear in bold in the table. It should be noted that the Maui has achieved its best results (31.6%) only after conducting manual analysis and feature selection on the dataset. Whereas, in our algorithm, feature selection is an automated and integral part of the learning process, which is achieved by adjusting the degree and weight parameters.

Table 3. Performance comparison with human annotators and rival machine annotators.

Method	Learning Approach	Number of Keyprases Assgined per document, *nk*	Avg. inter consistency with human annotators (%)		
			Min.	Avg.	Max.
TFIDF (baseline)	n/a - unsupervised	5	5.7	8.3	14.7
KEA++ (KEA-5.0)	Naïve Bayes	5	15.5	22.6	27.3
Grineva et al.	n/a - unsupervised	5	18.2	27.3	33.0
Maui	Naïve Bayes (all 14 features)	5	22.6	29.1	33.8
Maui	Bagging decision trees (all 14 features)	5	25.4	30.1	38.0
Human annotators (gold standard)	n/a - senior CS students	Varied, with an average of 5.7 per document	21.4	30.5	37.1
CKE	n/a - unsupervised	5	22.7	30.6	38.3
Current work	**n/a - unsupervised**	**5**	**19.1**	**30.7**	**37.9**
Maui	Bagging decision trees (13 best features)	5	23.6	31.6	37.9
Current work (LOOCV)	GA, threshold=800, unique bests method	5	12.3	32.8	58.1
Current work (LOOCV)	GA, threshold=200, unique bests method	5	13.9	32.9	56.7
Current work (LOOCV)	**GA, threshold=400, unique bests method**	**5**	**14.0**	**33.5**	**58.1**

5. Discussion

Using inter-indexer consistency scores to measure the quality of keyphrases assigned to the test documents by our machine annotator, in effect, defines the goodness of a given machine-generated keyphrase as its popularity among human annotators. Accordingly, a machine-generated keyphrase which is shared by most human annotators is given the highest quality score. This definition of quality addresses the common need of information retrieval systems such as retrieval engines, browsing interfaces, and classification and clustering engines for accurate and, more importantly, consistent subject metadata.

Table 4 shows the five most frequent keyphrases assigned by human annotators to two sample documents from the wiki-20 dataset and compares them with the top five keyphrases assigned to those documents by our supervised and unsupervised machine annotators and their rivals Maui and CKE. These two documents represent the two extreme cases in the dataset, where our machine annotators have yielded the highest and lowest inter-consistency scores. The number in parentheses appearing after each keyphrase, represents the number of human annotator teams who have assigned that keyphrase to the document. The keyphrases are sorted in a descending order according to their assignment frequency by human annotator teams or their keyphraseness score in case of machine annotators. Examining the second sample document (Doc. ID 25473) reveals two main factors behind the low inter-consistency score achieved on this document: (a) the low inter-consistency among human annotators themselves; (b) the fact that neither the descriptor nor any of the non-descriptors of the topic “content-based image retrieval”, which is the most popular topic among the human annotators, occur in the document.

Table 4. Top 5 Keyphrases assigned to two sample documents by human and machine annotators.

Doc. ID	Doc. Title	Human teams	Our supervised machine annotator	Maui	Our unsupervised machine annotator	CKE
10894 (ExpC$_5$)	A safe, efficient regression test selection technique	Regression testing (15)	Regression testing (15)	Algorithm (7)	Regression testing (15)	Regression testing (15)
		Software maintenance (13)	Algorithm (7)	Control flow (0)	Software testing (9)	Software maintenance (13)
		Control flow graph (10)	Software maintenance (13)	Software maintenance (13)	Algorithm (7)	Control flow graph (10)
		Software testing (9)	Software testing (9)	Computer software (1)	Computer software (1)	Software testing (9)
		Algorithm (7)	Computer software (1)	Test suite (2)	Control flow (0)	Test suite (2)
Avg. inter-consistency with human teams		**52.80%**	**58.10%**	**29.70%**	**41.30%**	**65.57%**
25473 (ExpC$_{15}$)	Extracting Multi-Dimensional Signal Features for Content-Based Visual Query	Content-based image retrieval (7)	Computer vision (5)	Tree (data structure) (1)	Computer vision (5)	computer vision (5)
		Image processing (5)	Wavelet (1)	Wavelet (1)	Color (0)	feature extraction (5)
		Computer vision (5)	Discrete cosine transform (2)	Discrete cosine transform (2)	Quadtree (2)	color histogram (2)
		Image compression (5)	Pattern recognition (2)	Database (2)	Discrete cosine transform (2)	indexing and retrieval (0)
		Feature extraction (5)	Information retrieval (1)	Color histogram (2)	Wavelet (1)	image compression (5)
Avg. inter-consistency with human teams		**15.30%**	**14.00%**	**10.10%**	**14.00%**	**16.09%**

As shown in Table 3, our unsupervised machine annotator outperforms its rivals and yields a performance on a par with the average performance of human annotators. Our supervised machine annotator also outperforms its rivals and maintains a performance considerably above the average performance of the human annotators. However, as shown in Table 5, the best human annotator (team 15) still outperforms our machine annotator with a large margin. Our unsupervised machine annotator and its rival CKE both outperform 5 out of 15 human teams. Whereas, our supervised machine annotator outperforms 12 of the human teams, and its rival, Maui, outperforms 11 teams. These results reaffirm the superiority of supervised machine annotators over the unsupervised ones in terms of the quality of resulting keyphrases. However, the unsupervised machine annotators maintain their appeal due to their lower computational footprint and “plug & play” nature.

Table 5. Performance comparison with human annotators and rival machine annotators.

TeamID/Machine	Native English speakers	Avg. study year	Avg. Inter-consistency (%) with (other) human annotators
1	No	4.5	21.4
2	No	1	24.1
3	No	4	26.2
4	No	2.5	28.7
5	Yes	4	30.2
CKE - unsupervised	**n/a**	**n/a**	**30.6**
Our unsupervised machine annotator	**n/a**	**n/a**	**30.7**
6	Mixed	4	30.8
7	Yes	3	31
8	No	3	31.2
9	Yes	4	31.6
10	Yes	3.5	31.6
11	Yes	4	31.6
Maui - bagging decision trees (13 best features)	**n/a**	**n/a**	**31.6**
12	Mixed	3	32.4
Our supervised machine annotator Threshold=400, unique bests method	**n/a**	**n/a**	**33.5**
13	Yes	4	33.8
14	Mixed	4	35.5
15	Yes	4	37.1

In order to ensure a fair and objective comparison between the performance of our method and other methods reported in the literature, we have fixed the number of keyphrases, *nk*, assigned by our machine annotator to 5 per test document (same as the rivals). However, it should be noted that in case of the test dataset used in this study, increasing the number of keyphrases assigned by our machine annotator per document from 5 to 6 yields an improvement in the overall inter-consistency score achieved.

The supervised learning approach to keyphrase annotation was first introduced by Turney [7]. Based on an extensive series of experiments, he concluded that in general a custom-designed learning algorithm similar to the one proposed in this work outperforms general-purpose ML algorithms, such as decision trees. However, since then, the great majority of reported supervised keyphrase annotation methods have adopted general-purpose learning algorithm, such as Naïve Bayes and bagging decision trees which have shown superior performance over other general-purpose ML algorithms used in keyphrase annotation tasks (e.g., see [13, 17, 44]). Surprisingly, support vector machines, which has been deployed with high success in numerous text classification works reported in the literature, has rarely been mentioned in the reported keyphrase annotation works. Our own preliminary experiments with applying SVM to keyphrase annotation tasks and those of others [44] show that the imbalanced nature of classes in the keyphrase annotation training data, where the great majority of learning instances (candidates) in the training documents belong to the negative class, i.e., are not suitable keyphrases, causes the SVM to underperform; and popular methods for dealing with imbalanced datasets, such as increasing the weight of positive instances (suitable candidates), or down-sampling the negative instances, do not significantly improve its performance.

A drawback of using genetic algorithms is that they can be computationally expensive in comparison to other learning algorithms, such as decision trees, and hence require much longer training time. In our supervised GA-based approach, the fitness of each individual, i.e., ranking function, in a given population is measured by applying it to the set of training documents and evaluating the results in terms of overall inter-consistency with human annotators. In case of ExpC in our first round of evaluation, where 19 documents are used for training, this process for each population of 40 individuals takes on average about 3 seconds, running on a mid-range PC. This translates to an average of about an hour training time, which, although much longer than that required for training decision trees, is reasonably affordable. However, the training time may be expected to raise significantly as the size of training dataset increases, which could be compensated for by reducing the population size or threshold as we expect the bigger training datasets to require a less number of evolutionary generations to find the optimum solution.

As expected and is evident from the results of the three evaluation rounds, the *unique bests* method used for applying the learnt models to the unseen documents, overall outperforms the *last best* method and yields more stable results. This method of model averaging is similar to bagging which is proven to enhance the performance of decision trees in keyphrase annotation tasks [7, 17, 44]. The *unique bests* method is similar to bagging in that it uses an ensemble of ranking functions instead of a single one to compute the overall keyphraseness score of a given candidate. However, unlike the bagging, each ensembled ranking function is tuned according to the whole training dataset rather than a subset of it.

Table 6 shows the average values for the GA adjusted weight parameters of the ranking functions (unique bests) for the LOOCV experiment (ExpC) in the first round of evaluation. This data indicates *first occurrence* (f_2) as the strongest feature and *translations count* (f_{20}) as the weakest. As expected, both semantic relatedness features *link-based relatedness to other topics* (f_{12}) and *category-based relatedness to other topics* (f_{14}) have been assigned high weights. Also, as expected and discussed in Section 2, *lexical unity* (f_7) outweighs *lexical diversity* (f_6), *generality* (f_{15}) outweighs *speciality* (f_{16}), and *links out ratio* (f_{18}) outweighs *links in ratio* (f_{19}). In addition, *occurrence spread* (f_4) and *first occurrence* (f_2) both outweigh *last occurrence* (f_3), which is in agreement with the results of a previous study on the wiki-20 dataset [17].

Table 6. Average weight parameter values of ranking functions (unique bests).

Dataset	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}	f_{16}	f_{17}	f_{18}	f_{19}	f_{20}
ExpC	1.11	1.54	0.72	1.03	0.55	0.82	1.06	1.47	1.11	0.59	0.60	1.14	0.66	1.21	1.18	0.90	0.73	1.20	0.91	0.32

Table 7 shows the average values for the degree parameters of the ranking functions. Since the degree values of ranking functions are averaged over all the unique best functions and also due to dependencies between the features, we can not interpret all the GA adjusted degree parameters. However, as expected and discussed in Section 3, in great majority of cases *term frequency* (f_1) is scaled logarithmically to compensate for the word burstiness phenomenon. In addition, *First occurrence* (f_2) and *occurrence spread* (f_4) both have been assigned the highest degree values compared to the other features, which means they have a significant effect on the overall keyphraseness scores of candidates only when holding large values, i.e., when the candidate is well spread in the document and/or occurs very close to the beginning of the document. Similar to f_2 and f_4, the *last occurrence* (f_3) should be deemed significant only when holding large values. However, the data shows that f_3 has been scaled logarithmically in a majority of cases, which could mean the GA has repurposed it as a noisy duplicate of *first occurrence* in case of candidates occurring close to the beginning of the document with low *occurrence spread* values. In case of the semantic relatedness features, *category-based relatedness to other topics* (f_{14}) has been assigned higher degree values than *link-based relatedness to other topics* (f_{12}) which could mean the latter is generally more reliable. It should be noted that both weight and degree parameters of features would vary from one dataset to another due to their underlying differences as discussed in Section 3.

Table 7. Average degree parameter values of ranking functions (unique bests).

Dataset	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}	f_{16}	f_{17}	f_{18}	f_{19}	f_{20}
ExpC	0.5	1.53	0.59	1.4	1.22	1.07	1.05	0.36	0.25	0.59	0.74	0.92	0.72	1.37	1.24	1.09	1.23	0.65	0.64	0.84

6. Conclusion and Future Work

In this article, we introduced an unsupervised and a supervised GA-based keyphrase annotation method for scientific literature utilizing a large number of features derived from Wikipedia. We evaluated the performance of both methods in terms of the consistency of their resulted keyphrases for a collection of test documents with those assigned by human annotators. The results of the three rounds of evaluation show that both methods outperform their rivals and yield a performance above the average performance of human annotators in terms of overall inter-indexer consistency.

Despite its superior performance, the main limitation of the proposed GA-based approach is its rather long training time in comparison to general-purpose ML algorithms such as decision trees. This limitation may be addressed in future versions by parallelizing the process of evaluating the fitness of individual ranking functions in each generation, which is supported in ECJ via distributed evaluation functionality. This enhancement allows us to conduct a large number of complementary experiments efficiently in order to examine the effect of various GA settings, such as population size, min/max gene, crossover type, mutation type, and mutation probability on the overall performance. A number of new features, such as occurrence positions of candidates in respect to the logical structure of the given document, as reported in [9], would be added and evaluated in the future version. The evaluation of the new version would also include an extra dataset called FAO-30 [17] which like the wiki-20 dataset is annotated by multiple human annotators independently and allows us to accurately evaluate the performance of our machine annotator using the inter-indexer consistency measure.

Acknowledgments

We would like to thank David Milne for developing the Wikipedia-miner, making it publicly available as open-source software, and the technical assistance he has given us with this toolkit. We also thank Alyona Medelyan and Ian Witten for the wiki-20 dataset.

References

[1] Hunter L. and Cohen K. B., Biomedical Language Processing: What's Beyond PubMed?, Molecular Cell 2006; 21, 5: 589-594.

[2] Wu Y.-f. and Li Q., Document keyphrases as subject metadata: incorporating document key concepts in search results, Information Retrieval 2008; 11, 3: 229-249.

[3] Jones S. and Paynter G. W., Automatic extraction of document keyphrases for use in digital libraries: Evaluation and applications, Journal of the American Society for Information Science and Technology 2002; 53, 8: 653-677.

[4] Grineva M., Grinev M. and Lizorkin D. Extracting key terms from noisy and multi-theme documents. In: 18th international conference on World wide web; 2009; Madrid, Spain: ACM; 2009.

[5] Mahdi A. E. and Joorabchi A., A Citation-based approach to automatic topical indexing of scientific literature, Journal of Information Science 2010; 36, 6: 798-811.

[6] Witten I. H., Paynter G. W., Frank E., Gutwin C. and Nevill-Manning C. G. KEA: practical automatic keyphrase extraction. In: fourth ACM conference on Digital libraries; 1999; Berkeley, California, United States: ACM; 1999.

[7] Turney P. D., Learning Algorithms for Keyphrase Extraction, Inf. Retr. 2000; 2, 4: 303-336.

[8] Turney P. D. Coherent keyphrase extraction via web mining. In: Proceedings of the 18th international joint conference on Artificial intelligence; 2003; Acapulco, Mexico: Morgan Kaufmann Publishers Inc.; 2003. p. 434-439.

[9] Nguyen T. D. and Kan M.-Y. Keyphrase extraction in scientific publications. In: Proceedings of the 10th international conference on Asian digital libraries: looking back 10 years and forging new frontiers; 2007; Hanoi, Vietnam: Springer-Verlag; 2007. p. 317-326.

[10] Markó K. G., Hahn U., Schulz S., Daumke P. and Nohama P. Interlingual Indexing across Different Languages. In: Computer-Assisted Information Retrieval (Recherche d'Information et ses Applications) - RIAO; 2004. p. 82-99.

[11] Pouliquen B., Steinberger R. and Ignat C. Automatic annotation of multilingual text collections with a conceptual thesaurus. In: Ontologies and Information Extraction. Workshop at EUROLAN'2003; 2003.

[12] Medelyan O. and Witten I. H. Thesaurus based automatic keyphrase indexing. In: Proceedings of the 6th ACM/IEEE-CS joint conference on Digital libraries; 2006; Chapel Hill, NC, USA: ACM; 2006. p. 296-297.

[13] Medelyan O. and Witten I. H., Domain-independent automatic keyphrase indexing with small training sets, Journal of the American Society for Information Science and Technology 2008; 59, 7: 1026-1040.

[14] Milne D., Medelyan O. and Witten I. H. Mining Domain-Specific Thesauri from Wikipedia: A Case Study. In: Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence; 2006: IEEE Computer Society; 2006. p. 442-448.

[15] Medelyan O., Milne D., Legg C. and Witten I. H., Mining meaning from Wikipedia, Int. J. Hum.-Comput. Stud. 2009; 67, 9: 716-754.

[16] Medelyan O., Witten I. H. and Milne D. Topic Indexing with Wikipedia. In: first AAAI Workshop on Wikipedia and Artificial Intelligence (WIKIAI'08); 2008; Chicago, US: AAAI Press; 2008.

[17] Medelyan O., Human-competitive automatic topic indexing (Ph.D Thesis, University of Waikato, New Zealand, 2009). Available at: http://adt.waikato.ac.nz/public/adt-uow20091029.160923 (accessed 11 March 2012)

[18] Milne D. An open-source toolkit for mining Wikipedia. In: New Zealand Computer Science Research Student Conference; 2009.

[19] Salton G. and Buckley C., Term-weighting approaches in automatic text retrieval, Inf. Process. Manage. 1988; 24, 5: 513-523.

[20] Csomai A. and Mihalcea R. Linguistically Motivated Features for Enhanced Back-of-the-book Indexing. In: Proceedings of the Association for Computational Linguistics (ACL 2008); 2008 June 2008; Columbus, Ohio; 2008.

[21] Hulth A., Combining Machine Learning and Natural Language Processing for Automatic Keyword Extraction (Ph.D Thesis, Stockholm University, 2004). Available at: http://people.dsv.su.se/~hulth/thesis_hulth.pdf (accessed 11 March 2012)

[22] Kim S. N. and Kan M.-Y. Re-examining automatic keyphrase extraction approaches in scientific articles. In: Proceedings of the Workshop on Multiword Expressions: Identification, Interpretation, Disambiguation and Applications; 2009; Suntec, Singapore: Association for Computational Linguistics; 2009. p. 9-16.

[23] Barker K. and Cornacchia N. Using Noun Phrase Heads to Extract Document Keyphrases. In: Proceedings of the 13th Biennial Conference of the Canadian Society on Computational Studies of Intelligence: Advances in Artificial Intelligence; 2000: Springer-Verlag; 2000. p. 40-52.

[24] enwiki dump progress on 22/07/2011, (Wikimedia dump service, 2011), http://dumps.wikimedia.org/enwiki/20110722/ (accessed 11 March 2012)

[25] Porter M. F., An algorithm for suffix stripping, Program 1980; 14, 3: 130−137.

[26] M.F.Porter, The English (Porter2) stemming algorithm, (Snowball, 2002), http://snowball.tartarus.org/algorithms/english/stemmer.html (accessed 11 March 2012)

[27] Milne D. and Witten I. H. Learning to link with wikipedia. In: Proceedings of the 17th ACM conference on Information and knowledge management; 2008; Napa Valley, California, USA: ACM; 2008. p. 509-518.

[28] Milne D. and Witten I. H. An effective, low-cost measure of semantic relatedness obtained from Wikipedia links. In: first AAAI Workshop on Wikipedia and Artificial Intelligence (WIKIAI'08); 2008; Chicago, I.L; 2008.

[29] Strube M. and Ponzetto S. P. WikiRelate! computing semantic relatedness using wikipedia. In: proceedings of the 21st national conference on Artificial intelligence - Volume 2; 2006; Boston, Massachusetts: AAAI Press; 2006. p. 1419-1424.

[30] Salah A. A., Gao C., Suchecki K. and Scharnhorst A., Need to Categorize: A Comparative Look at the Categories of Universal Decimal Classification System and Wikipedia, Leonardo 2012; 45, 1: 84-85.

[31] Rada R., Mili H., Bicknell E. and Blettner M., Development and application of a metric on semantic nets, Systems, Man and Cybernetics, IEEE Transactions on 1989; 19, 1: 17-30.

[32] Leacock C. and Chodorow M. Combining local context and WordNet similarity for word sense identification. WordNet: An Electronic Lexical Database. In C. Fellbaum (Ed.), MIT Press, 1998, p. 265-283.

[33] O'Madadhain J., Fisher D., Nelson T., White S. and Boey Y.-B., JUNG 2.0, (Released under the open source GPL licence, 2009), http://jung.sourceforge.net/index.html (accessed 11 March 2012)

[34] Bastian M., Heymann S. and Jacomy M., Gephi: An Open Source Software for Exploring and Manipulating Networks (2009).

[35] Jones K. S., IDF term weighting and IR research lessons, Journal of Documentation 2004; 60, 5: 521-523.

[36] Church K. W. and Gale W. A., Poisson Mixtures, Journal of Natural Language Engineering 1995; 1, 2: 163-190.

[37] Luke S., Panait L., Balan G., Paus S., Skolicki Z., Popovici E., Sullivan K., Harrison J., Bassett J., Hubley R., Chircop A., Compton J., Haddon W., Donnelly S., Jamil B., Zelibor J., Kangas E., Abidi F., Mooers H. and O'Beirne a. J., ECJ - A Java-based Evolutionary Computation Research System, (George Mason University's Evolutionary Computation Laboratory), http://cs.gmu.edu/~eclab/projects/ecj/ (accessed 11 March 2012)

[38] Medelyan O. and Witten I. H., wiki-20 dataset, (University of Waikato, New Zealand, 2009), http://maui-indexer.googlecode.com/files/wiki20.tar.gz (accessed 11 March 2012)

[39] Rolling L., Indexing consistency, quality and efficiency, Information Processing & Management 1981; 17, 2: 69-76.

[40] Rijsbergen C. J. V., Information retrieval (Butterworths, London / Boston, 1979).

[41] Medelyan O., Frank E. and Witten I. H. Human-competitive tagging using automatic keyphrase extraction. In: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3 - Volume 3; 2009; Singapore: Association for Computational Linguistics; 2009. p. 1318-1327.

[42] Zunde P. and Dexter M. E., Indexing consistency and quality, American Documentation 1969; 20, 3: 259-267.

[43] Giles C. L., Kurt D. B. and Steve L. CiteSeer: an automatic citation indexing system. In: Proceedings of the third ACM conference on Digital libraries; 1998; Pittsburgh, Pennsylvania, United States: ACM; 1998.

[44] Lopez P. and Romary L. HUMB: Automatic key term extraction from scientific articles in GROBID. In: Proceedings of the 5th International Workshop on Semantic Evaluation; 2010; Los Angeles, California: Association for Computational Linguistics; 2010. p. 248-251.