SemCluster: Unsupervised Automatic Keyphrase Extraction Using Affinity Propagation

Conference Paper *in* Advances in Intelligent Systems and Computing · September 2017
DOI: 10.1007/978-3-319-66939-7_19

2 authors:

H. Alrehamy
Cardiff University
6 PUBLICATIONS **17** CITATIONS
SEE PROFILE

Coral Walker
Cardiff University
14 PUBLICATIONS **36** CITATIONS
SEE PROFILE

SemCluster: Unsupervised Automatic Keyphrase Extraction Using Affinity Propagation

Hassan H. Alrehamy and Coral Walker(✉)

School of Computer Science and Informatics, Cardiff University, Cardiff, UK
{alrehamy,huangy}@cardiff.ac.uk

Abstract. Keyphrases provide important semantic metadata for organizing and managing free-text documents. As data grow exponentially, there is a pressing demand for automatic and efficient keyphrase extraction methods. We introduce in this paper *SemCluster*, a clustering-based unsupervised keyphrase extraction method. By integrating an internal ontology (i.e., WordNet) with external knowledge sources, *SemCluster* identifies and extracts semantically important terms from a given document, clusters the terms, and, using the clustering results as heuristics, identifies the most representative phrases and singles them out as keyphrases. *SemCluster* is evaluated against two baseline unsupervised methods, TextRank and KeyCluster, over the Inspec dataset under an F1-measure metric. The evaluation results clearly show that *SemCluster* outperforms both methods.

Keywords: Keyphrase extraction · Clustering-based AKE · Unsupervised AKE

1 Introduction

Keyphrases provide a concise description of the essential meaning of a document and play a vital role in many information management tasks. Currently, only a small fraction of digital documents have assigned keyphrases. This is because keyphrase assignment is done manually in most cases, and such an undertaking is laborious and time-consuming. In the booming era of Big Data, automating the task of keyphrase extraction has gained significant attention and research effort. Automatic Keyphrase Extraction (AKE) is a natural language processing (NLP) task concerned with the automatic selection of representative phrases from the body of a document [1]. AKE approaches may be divided into two categories [2]: supervised and unsupervised. In a supervised approach, AKE is cast as a classification problem, and a classifier is employed to determine whether a candidate phrase in a document is a keyphrase. Although many current supervised AKE approaches deliver promising results, they require careful selection of manually annotated training datasets, which itself is tedious and may lead to inconsistency, especially in a heterogeneous processing environment that requires cross-domain tractability [15]. Unsupervised AKE is a much recent trend aimed at discovering the underlying structure of a document without the assistance of machine

F. Chao et al. (eds.), *Advances in Computational Intelligence Systems*,
Advances in Intelligent Systems and Computing 650, DOI 10.1007/978-3-319-66939-7_19

learning, and thus reducing expensive human labour by using various extraction techniques. A typical unsupervised approach uses graph-based modelling [3–5], where the document is modelled as a graph, terms as nodes, and relations between nodes as weighted edges. A typical workflow is: (i) weight importance of each node based on its relations to other nodes, (ii) rank each node based on its weight, (iii) select nodes of top rank as candidates. However, an unsupervised approach may encounter the following challenges regarding keyphrase extraction: firstly, a top-ranking term of statistical and/or semantic importance is not guaranteed to be associated with keyphrases representative of the document theme [6]; secondly, a term representative of the document theme is not guaranteed to be a top-ranking term if it occurs infrequently in the document [2]. In the text segment in Fig. 1, "*Olympics*" appears frequently, so we are not surprised to see many AKEs select "*Olympics*", "*Olympic games*", and "*Olympic movement*" as keyphrases without considering that "*Olympic movement*" has no immediate relevance to the document theme (i.e., "*athelete news*") and is an unsuitable keyphrase. Including an irrelevant phrase as one of the keyphrases for a document can cause confusion in search queries. For instance, a miscellany of documents about "*nonprofit organization*" or "*regulations*" may be returned in response to a search query for documents about "*athlete news*". On the other hand, "*dash*", a term of significant semantic importance, is rarely identified as a keyphrase ("*100-m dash*") by many AKEs because of its infrequent occurrence in the document.

Canadian **Ben Johnson** left the **Olympics** today "in a complete state of shock," accused of cheating with drugs in the worlds fastest **100-meter dash** and stripped of his **gold medal**. The prize went to American **Carl Lewis**. Many athletes accepted the accusation that Johnson used a muscle-building but dangerous and illegal anabolic steroid called **Stanozolol** as confirmation of what they said they know has been going on in track and field. Two tests of Johnsons urine sample proved positive and his denials of **drug use** were rejected today. "This is a blow for the **Olympic Games** and the Olympic movement," said International Olympic Committee President Juan Antonio Samaranch.

Fig. 1. A news article on "Ben Johnson" from the *DUC-2001 dataset*.

In this paper, we introduce *SemCluster* to address the above challenges. *SemCluster* is an unsupervised clustering-based method for effective keyphrase extraction in any domain. Motivated by [2,6], *SemCluster* performs clustering on the terms of a given document. It groups similar terms within the same cluster based on their lexical, semantic, and statistical information, and each resulting cluster may implicitly represent a topic of the document. Terms that are close to the centroids of specific clusters are selected as seeds to ensure finding candidate phrases in the document that semantically cover its main theme. Finally, *SemCluster* refines the candidate phrases and the resulting candidates are chosen as keyphrases.

The rest of the paper is organized as follows: a summary of related work is presented in Sect. 2, the *SemCluster* details are provided in Sect. 3. In Sect. 4, we undertake the evaluation and analysis of *SemCluster* performance, and, finally, we conclude the work and discuss future work in Sect. 5.

2 Related Work

A plethora of approaches for unsupervised keyphrase extraction have been devised, of which the simple ones rely fully on frequency statistics, such as TF-IDF, while others explore more sophisticated techniques. Steier and Belew [7] use mutual information statistics to discover bi-gram keyphrases in a document. However, the keyphrases it identifies can be of unpredictable length. Barker and Cornacchia [8] propose a simple system that selects only noun phrases as keyphrases without considering that some nouns carry minimal semantic significance. Litvak and Last [9] propose HITS, a graph-based ranking algorithm that extracts keyphrases from scientific documents and selects top-ranking vertices of the graph as keyphrases. Mihalcea and Tarau [5] present TextRank, a graph-based algorithm that uses the concept of Voting [10] between keywords and selects those with the most votes as keyphrases. Tsatsaronis et al. [11] present SemanticRank, a graph-based algorithm similar to TextRank, where edges are weighted based on their semantic relatedness, and nodes linked to heavily weighted edges are considered to be keyphrases. SemanticRank was tested using the *Inspec abstracts dataset*, and according to its authors has better performance than many other variations of PageRank and HITS. Bracewell et al. [12], on the other hand, propose a model that extracts noun phrases from a document, groups them into clusters in which two noun phrases are considered close if they share one or more noun terms, and, depending on their frequencies, ranks the clusters and selects the top ones in each clusters as keyphrases. Finally, KeyCluster is a similar clustering-based method proposed by Liu et al. [6]. It extracts single noun terms, groups them into clusters based on their semantic relatedness using Wikipedia and co-occurrence similarity measures, and selects phrases that contain one or more cluster centroids, and that follow a certain linguistic pattern, as keyphrases. KeyCluster outperforms many prominent AKE methods but suffers several practical drawbacks. Clustering-based methods, in general, cannot guarantee that all generated clusters are important in representing the document theme, and selecting the centroid of an unimportant cluster as a heuristic to identify and extract keyphrases leads to inappropriate keyphrases.

3 SemCluster Overview

With O as its ontology, *SemCluster* takes a free-text document D as input. Using O, *SemCluster* extracts from D keyphrases that are most representative of the theme of D. The *SemCluster* workflow is detailed in the following subsections.

3.1 Candidate Term Extraction and Disambiguation

The Term Extraction and Disambiguation stage is the first step of *SemCluster*. It performs text preprocessing on D involving tokenization, sentence boundary detection, part-of-speech (POS) tagging, and chunking. Penn Treebank notion is adopted for chunking and Part-Of-Speech (POS) tagging. The aim of chunking is to group words into chunks based on their discrete grammatical meanings. It is observed that the majority of manually-assigned keyphrases are typically embedded in noun phrases (NP chunks) [6,8,13]. Thus, *SemCluster* considers only NP chunks to find keyphrases, and detects and extracts terms in each NP chunk based on their POS annotations. We initially allow the selection of n-gram terms (where $0 < n \leq 5$) using the following POS patterns:

Table 1. Candidate term POS extraction patterns with examples from Fig. 1.

Pattern	Example
$\mathtt{N} = (\mathrm{NN}\|\mathrm{NNS})$	`dash/NN`, `prize/NN`, `drugs/NNS`
$\mathtt{C} = (\mathrm{JJ}) * (\mathrm{NN}\|\mathrm{NNS})+$	`anabolic/JJ steroid/NN`, `gold/NN medal/NN`
$E = (\mathrm{NNP}\|\mathrm{NNPS}) * (\mathcal{S}) * (\mathrm{NNP}\|\mathrm{NNPS})+$	`Stanozolol/NNP`, `Ben/NNP Johnson/NNP`, `Olympic/NNP Games/NNPS`

In Table 1, $\mathcal{N}$ denotes *Noun*, a singleton word tagged as (NN) or (NNS). $\mathcal{C}$ denotes *Compound Noun*, a sequence of words starting either with an adjective (JJ) or noun (NN and NNS). E denotes *Entity*, a sequence of words of proper nouns (NNP or NNPS) with at most one stop-word ($\mathcal{S}$), such as *the* and *of*, in the middle. Each term extracted using these patterns is mapped into *SemCluster*'s ontology O, and, depending on the mapping result, a term is regarded either as a *candidate term* or *miscellaneous*. When a term does not map to any entries in the ontology, it is decomposed into smaller constituents to be mapped again. The terms that fail to find matches even after being reduced to smaller constituents are discarded.

SemCluster's core ontology is WordNet [14], a widely used lexical database. WordNet has four lexical categories: nouns, verbs, adjectives and adverbs, and we use only the noun category. Nouns of equivalent meaning are grouped into synsets. Each synset consists of a list of synonyms and its defining gloss. Synsets are connected to other synsets by means of semantic relations. Synsets are organised into hyponym/hypernym (Is-A), and meronym/holonym (Part-Of) relationships, providing a hierarchical tree-like structure.

In practice, no knowledge base is comprehensive, and neither is WordNet. WordNet contains a *limited* number of English nouns collected over a decade ago and does not support newly emerged nouns. To compensate for this, we design a novel procedure to integrate external ontology-based knowledge bases, such as DBPedia,[1] Yago,[2] and other *ad hoc* ontologies. The workflow of the

[1] http://www.dpedia.org.
[2] http://yago-knowledge.org.

proposed procedure is as follows: for a required external knowledge base, its schema is modeled as an external ontology, and each entry in the knowledge base is assigned to one or more type classes. To perform a meaningful integration, the external ontology is horizontally aligned with *SemCluster*'s core ontology by mapping each type class to its exact or semantically equivalent synset in the core ontology. Such alignment is a one-to-one mapping where each class is mapped to exactly one synset. When *SemCluster* encounters a term that does not exist in the core ontology, the term is queried against the external knowledge base(s). If matches are found, each matching entry is retrieved from the knowledge base and considered as an external contextual meaning (or *sense*) for the given term. All external ontology classes associated with external senses are mapped into their corresponding WordNet synsets and are considered as hypernyms. The synset that corresponds to the deepest class in the external ontology's hierarchy is considered the direct hypernym of the external sense. With this construct, we allow *SemCluster* to dynamically generate appropriate senses for new terms that are absent in WordNet, and expand the set of synsets for an existing term. To illustrate, we consider integrating DBPedia and aligning all its schema classes[3] with their equivalent WordNet synsets. For instance, the class "*dbp:Athlete*" in DBPedia is directly mapped to "*wn:Athlete#n1*" in WordNet, while "*dbp:MusicFestival*" is mapped to its equivalent synset "*wn:Fete#n2*". Revisiting Fig. 1, we see that the entity "*Ben Johnson*" has no entries in WordNet but five entries in DBPedia. Accordingly, *SemCluster* generates five new senses for "*Ben Johnson*", each matching one entry in DBPedia. The third sense, ("*Ben Johnson (Sprinter)*"), is associated with four classes: "*owl:Thing*", "*dbo:Agent*", "*dbo:Person*", "*dbo:Athlete*". The deepest among the four classes, "*dpo:Athlete*", becomes the hypernym of the third sense and is referred to as "*wn:Athlete#n1*". After mapping each extracted term against the extended ontology $O^{\dagger}$, only a subset of the terms are regarded as candidate terms. We denote the set of the candidate terms by T_D. Due to pattern-based term extraction, especially when D contains informal text, T_D may harbour noisy terms that can adversely affect similarity computation and clustering performance. Noisy terms are nouns with no semantic value (e.g. "*one*", "*someone*"). To identify and remove noisy terms, *SemCluster* maps each term in T_D to a list of the most frequent noisy terms in the English language, and any term found in the list is removed from T_D.

Prior to semantic similarity computation, *SemCluster* must identify the contextual meaning (or *sense*) of each term in T_D. Word Sense Disambiguation (WSD) is an NLP task that concerns giving machines the ability to computationally determine which sense of a term is activated by its use in a particular context. WSD approaches are generally divided into three categories [16]: supervised, unsupervised, and knowledge-based. *SemCluster* employs the SenseRelate-TargetWord algorithm [17] for term sense disambiguation. SenseRelate-TargetWord is a WordNet-based algorithm implemented in WordNet::Similarity, a widely used package in computational linguistics.

[3] DBpedia schema is available at http://mappings.dbpedia.org/server/ontology/classes/.

SenseRelate-TargetWord takes one *target* candidate term as input and outputs a synset for that term based on information about the target candidate term and a few other candidate terms surrounding the target. The surrounding candidate terms are called the context window. Let t_i be a target candidate term, $t_i \in T_D$, let the size of the context window be N, and let the set of surrounding candidate terms in the context window be W, $W = \{w_1, w_2, \ldots, w_N\}$, where, if $|w| < N$, then $N = |W|$. Each t_i has one or more senses, and each sense is assigned to a different synset in WordNet, and thus we denote the senses of the term t_i by $Sense(t_i) = \{s_{i1}, s_{i2}, \ldots, s_{im}\}$, where m is the number of related synsets. Through corresponding synsets, we obtain not only the synonym list and defining gloss of the synset specifying s_{ij}, but also the synonym lists and glosses of other synsets that are related to the synset and its sense s_{ij} via the following set of semantic relations: {Hypernym, Hyponym, Meronym, Holonym}. The goal of the algorithm is to find the synset responsible for s_{ij} whose synonyms and gloss content maximises the string-based overlap score for each w_k in the context window.

3.2 Candidate Terms Similarity Computation and Clustering

After disambiguating all candidate terms in T_D, we assume each $t_i \in T_D$ is associated with the specific information: a POS tag, its position in the document, and a pointer pointing correctly to a synset s_i associated with t_i. Once the synsets of candidate terms are determined, *SemCluster* computes the pairwise semantic similarity between candidate terms based on their synset pointers. There exist many similarity measures between synsets and they may be devided into three categories [18]: path-length based, information-content based, and feature based. Unlike the other two, path-length measures offer greater flexibility to compute the similarity between synsets based on *SemCluster*'s integrated ontology. The WuPalmer measure [19] is a prominent path-length measure to compute semantic similarity between two synsets s_i, s_j by finding the shortest path between each synset and the deepest common parent synset (*Least Common Subsumer* (LCS)). The similarity $S(s_i, s_j)$ is quantified by counting the nodes in the shortest path relative to LCS depth in the ontology hierarchy. The measure is given as [19]:

$$S(s_i, s_j) = \frac{2d}{L_{s_i} + L_{s_j} + 2d} \tag{1}$$

where d is the depth of LCS from the root node, L_{s_i} is the path length from s_i to LCS, and L_{s_j} is the path length from s_j to LCS. We modify the WuPalmer algorithm to capture extra semantic similarity between s_i and s_j. Path length measures in general, and WuPalmer in particular, focus on measuring synset similarities by exploiting the explicit semantic relations existing between them. However, WordNet does not cover all possible relations that may exist between synsets. For example, there is no direct link between "*wn:Bush#n4*" and "*wn:President#n2*", although they are clearly related if they co-occur in a document. To capture

explicit as well as implicit semantic similarities between s_i and s_j, we extend the WuPalmer measure as follows:

$$S(s_i, s_j) = \frac{2d + Overlap(C(s_i), C(s_j))}{L_{s_i} + L_{s_j} + 2d + Overlap(C(s_i), C(s_j))} \tag{2}$$

where $C(s_i)$ and $C(s_j)$ are functions that retrieve s_i and s_j information from WordNet in string format, and $Overlap(C(s_i), C(s_j))$ is a function that measures the string-based overlap between $C(s_i)$ and $C(s_j)$. Let $Synonyms(s_i)$ be a function that retrieves all the words in the synonym list of the synset s_i, $Gloss(s_i)$ be a function that retrieves the definition of s_i, $Related(s_i)$ be a function that retrieves the synonyms and definitions of all synsets connected directly to s_i via the relation set $\{Hypernym, Hyponym, Meronym, Holonym\}$, then $C(s_i)$ is defined as follows:

$$C(s_i) = Synonyms(s_i) \cup Gloss(s_i) \cup Related(s_i) \tag{3}$$

where $\cup$ is the string concatenation function. $Overlap(C(s_i), C(s_j))$ finds the maximum number of words shared in the output of $C(s_i)$ and $C(s_j)$ normalized by natural logarithm to prevent too great an effect of implicit semantic similarity on the WuPalmer explicit semantic similarity measurement. Thus, we define *Overlap* as follows:

$$Overlap(C(s_i), C(s_j)) = \log\left(C(s_i) \cap C(s_j) + 1\right) \tag{4}$$

The extended WuPalmer measure is used to compute the pairwise similarities between each pair of terms in T_D, and the result is a complete adjacency similarity matrix of size $|T_D| \times |T_D|$ denoted by A. Once we have produced A, we move on to the second phase of the step - terms clustering. There are many state-of-the-art clustering algorithms to cluster T_D efficiently. Affinity Propagation (AP) [20] is proposed as a powerful technique for exemplar learning by passing messages between nodes. It is reported to find clusters with much lower error compared to other algorithms and does not require specifying the number of desirable clusters in advance. Both merits are extremely important for *SemCluster* to support fully automated keyphrase extraction and hence AP is used as the clustering algorithm in *SemCluster*. The input to AP is the matrix A. The set T_D is modelled as a graph with nodes t_i, $t_i \in T_D$. An edge between t_i and t_j exists if $S(t_i, t_j) > 0$ and the weight of the edge is given by element $A[i][j]$. Initially, all the nodes are viewed as exemplars, and after a large number of real-valued information messages have been transmitted along the edges of the graph, a relevant set of exemplars and corresponding clusters is identified. In AP terms, the similarity $S(t_i, t_j)$ indicates how much t_j is suitable to be exemplar of t_i. In *SemCluster*, $S(t_i, t_j) = A[i][j]$, $i \neq j$. If there is no heuristic knowledge, self-similarities are called *preferences*, and are set as constant values. In *SemCluster*, the preference $S(t_i, t_i)$ is computed using the median. In AP two kinds of messages are exchanged between nodes: *responsibility* and *availability*. A responsibility message is sent from node t_i to candidate exemplar t_j, and

reflects the accumulated evidence for how well-suited t_j is to serve as the exemplar for t_i. An availability message is sent from candidate exemplar t_j to t_i, and reflects the accumulated evidence for how well-suited it would be for t_i to choose t_j as its exemplar. At the beginning, all availabilities are initialised to zero, and during m iterations, both responsibility and availability messages are updated iteratively until they remain constant for a specific number of iterations, and then both responsibilities and availabilities are combined to discover exemplars [20]. Eventually, every term in T_D is annotated with its exemplar. The number of clusters and other clustering information are directly obtained by grouping terms based on their shared exemplars. At start-up, the input set T_D can be *redundant* to reflect not only the semantic and lexical information of each term t_i, but also the influence of its frequency information on the clustering results, such that, if the term t_i is highly frequent in the document, its frequency can be a reason to qualify as an exemplar on the condition that t_i is always allocated the same sense s_{ij} in all its occurrences in D. Typically, clustering-based AKE approaches use cluster centroids as seeds [6,12], and any phrase in D containing one or more centroids is chosen as a keyphrase. From our empirical observation, we suggest that direct selection of centroids resulting from AP or similar algorithms may lead to poor keyphrase extraction recall and/or precision, due to the following two reasons:

Theme-Independent Seed Selection. Clustering-based methods assign equal importance to all cluster centroids [2]. Thus, a phrase containing a centroid of an unimportant cluster is ranked exactly equivalent to a phrase containing a centroid of an extremely important cluster relative to the document theme [21]. Consequently, there is no guarantee that the extracted keyphrases are the best representative phrases. Our solution to this is to discard irrelevant or marginally related clusters and keep the most relevant ones. The solution is largely based on the observation that clusters that sufficiently cover the document theme tend to be semantically more related to each other than irrelevant or marginally related clusters. Regarding AP, the exemplar is the best representative of its clusters semantics. Therefore, we assess the average of semantic relatedness strength of each exemplar against all other exemplars, and any cluster whose exemplar exhibits *weak* semantic relatedness is removed. Let C_D be the set of clusters resulting from clustering T_D, $C_D = \{C_1, C_2, \ldots, C_N\}$ where $N = |C_D|$. For each cluster C_i, we compute its exemplar's average semantic relatedness, $Ave(\varepsilon_i)$, as follows:

$$Ave(\varepsilon_i) = \frac{\Sigma_{i \neq j} SR(\varepsilon_i, \varepsilon_j)}{N - 1}, \quad N > 1 \tag{5}$$

In the above definition, $SR(\varepsilon_i, \varepsilon_j)$ measures the semantic relatedness between the exemplars of two clusters C_i, C_j. Each cluster C_i is ranked based on its exemplar average score, and it will be removed from C_D if its average score, $Ave(\varepsilon_i)$, is below the average of all clusters. $SR(\varepsilon_i, \varepsilon_j)$ is similar to $S(\varepsilon_i, \varepsilon_j)$ and measures the semantic relatedness between ε_i and ε_j. For instance, the terms "*drug*" and "*Olympics*" are not similar, but, because of their tendency to occur together ("*drug use*" appears frequently in "*Olympics*" themes), they are judged

to be semantically related. To quantify such relatedness in an unsupervised cross-domain environment, we are expanding *SemCluster* to take advantage of *Wikipedia*, the largest and fastest growing knowledge base. There is a selection of approaches that measure semantic relatedness by exploiting Wikipedia. *Explicit Semantic Analysis* [22] is one of the most accurate Wikipedia-based measures that, to a great extent, comes close to the accuracy of a human [23] and, hence, is employed by *SemCluster* to compute semantic relatedness.

Search Space Restriction. The sole reliance on the clusters centroids may lead to restricting the search space of finding the best representative phrases in a given document and, consequently, result in degrading keyphrase extraction recall and/or precision. Suppose we have a valid keyphrase containing a term t_j that is semantically close to a centroid term ε_i. The phrase will not be extracted simply because t_j is not a centroid. This may explain why spectral clustering outperforms AP clustering in KeyCluster experiments - the former allows multiple terms close to a cluster centroid to be chosen as seeds and accordingly extends the keyphrase search space. Taking advantage of this observation, *SemCluster* expands the selection of seeds from AP clustering in a fashion similar to that of spectral clustering. Let C'_D be the final set of clusters resulting from clustering T_D using AP after centroid relatedness average ranking, where $C'_D \subseteq C_D$, $C'_D = \{C_1, C_2, \ldots, C_k\}$. For each cluster C_i, $i \leq k$, we select its exemplar ε_i as a seed. We regard each member t_j in C_i ($t_j \neq \varepsilon_i$) as an additional seed if $S(\varepsilon_i, t_j) \geq \tau$, where $S(\varepsilon_i, t_j)$ is the computed score stored in A from a previous step (see Sect. 3.1) and τ is a predefined distance threshold specifying how semantically close t_j should be to the centroid ε_i in order to qualify as a seed. We repeat this procedure for all the clusters in C'_D to obtain a set of appropriate seeds from the extended search space.

3.3 Candidate Phrase Extraction and Keyphrase Selection

After selection of the seeds, each chunk NP_i in D is scanned, and *SemCluster* extracts any sequence of words inside NP_i that satisfies two conditions: (i) containing a seed, (ii) matching an *Extraction POS Pattern* by the following rules. (i) If NP_i contains a seed extracted using an E-pattern, the seed is regarded as a candidate phrase. (ii) If NP_i contains a seed extracted using a $\mathcal{C}$-pattern, two cases are considered: if the seed starts with (JJ), the sequence matching pattern $(\mathcal{C}) * (\text{NN}|\text{NNS})+$ is extracted from NP_i; if the seed starts with (NN), the sequence matching pattern $(\text{JJ}) * (\mathcal{C})+$ is extracted from NP_i. (iii) Finally, if NP_i contains a seed extracted using a $\mathcal{N}$-pattern, the sequence matching pattern $(\text{JJ}) * (\mathcal{N})+$ is extracted from NP_i. Each extracted sequence using these POS patterns is called a candidate phrase. The final step in the *SemCluster* processing approach is to refine the set of extracted candidate phrases. The refining step starts by pruning redundant candidate phrases. Two or more candidate phrases may be semantically equivalent but exist in different forms. They may be synonymous phrases, e.g. both "*Olympics*" and "*Olympic Games*" belong to the same synset in WordNet; or adjective-synonymous phrases, e.g., the Wikipedia

article referring to "*Bernard Madoff*" contains phrases that share an important seed "*fraud*" such as "*financial fraud*", "*gigantic fraud*", "*massive fraud*". In this case, we keep the first occurring candidate phrase and remove the others. There is also the case of subphrases, as in the example of "*Johnson*" and "*Ben Johnson*". Both phrases contain "*Johnson*", so we keep the longer phrase, which is more specific, and discard the short one.

4 Evaluation and Results

To evaluate *SemCluster*, we use an aligned ontology resulting from integrating WordNet with the DBPedia schema.[4] The external knowledge sources that we exploit for semantics-related computations are DBPedia lookup-server[5] for extending the core ontology semantic coverage, and EasyESA[6] for ESA-based semantic relatedness measurement. *SemCluster* performs text preprocessing using OpenNLP[7] with models trained on huge collections in multiple domains. We use WordNet[8] with some slight modifications to accommodate our needs, including re-indexing synsets for faster access, modifying each synset's gloss to keep only nouns and adjectives, and lemmatizing all noun tokens.

The dataset used in *SemCluster* evaluation is the *Inspec dataset*,[9] which is frequently used in the evaluation of AKE methods [5,6,13]. This is a collection of abstracts of scientific papers from the *Inspec Database*, consisting of 2000 abstracts. Each abstract is represented by three files: *.abstr*, *.contr* and *.uncontr*. The file *.abstr* contains the actual text; *.contr* contains keyphrases restricted to a specific dictionary; and *.uncontr* contains keyphrases assigned by human experts. In Hulth's experiment [13], the proposed method was supervised, and the dataset was spilt into three partitions: 1000 abstracts for training, 500 for validation, and 500 for testing. TextRank [5] and KeyCluster [6] are unsupervised methods, and thus only the test partition was used in their evaluations. Since *SemCluster* is also unsupervised, we adopt a similar approach to [5,6], and use only the test partition to provide a precise comparison with the mentioned AKE methods. In the dataset, phrases that are not in the abstract, if regarded by the human expert as suitable, are stored in *.uncontr* as keyphrases. In our evaluation, we consider only keyphrases that actually occur in the abstract. An output keyphrase is considered to be valid if it is identical to, semantically equivalent to, or is a subphrase of, a manually assigned keyphrase in *.uncontr*.

We use the *Inspec dataset* as a benchmark to compare *SemCluster*'s performance with two AKE methods: TextRank [5], a baseline unsupervised method, and KeyCluster [6], a current state-of-the-art unsupervised clustering-based method. KeyCluster is implemented using three different algorithms: Hierarchal

[4] The alignment results are available at http://www.pdlab.io/semcluster/inspec.rar.
[5] The lookup code is available at https://github.com/dbpedia/lookup.
[6] ESA code is available at http://treo.deri.ie/easyesa/.
[7] http://opennlp.apache.org.
[8] Wordnet v.3.1 is available at https://wordnet.princeton.edu/wordnet/download.
[9] Hulth's dataset copy is available at http://pdllab.io/semcluster/inspec.rar.

Clustering (HC), Spectral Clustering (SC), and Affinity Propagation (AP). Due to the poor performance of HC reported in [6], we evaluate KeyCluster based on SC and AP, not HC. During method comparisons, only the best results under the best possible settings, if any, for a given approach are considered. The setting for KeyCluster-SC is that m, the predefined number of clusters, is $m = \frac{2}{3}n$, where $n = |D|$. For KeyCluster-AP, the maximum number of iterations is set to 1000, the damping factor is set to 0.9, and the clustering preference is computed using *max*. *SemCluster* performs AKE in fully automatic mode. However, it requires tuning a set of parameters, which are: WSD context window size N (see Sect. 3.1), τ distance threshold (see Sect. 3.2), and AP algorithm default parameters (i.e., the maximum number of iterations M, damping factor λ, convit, and preference). From empirical observation, *SemCluster* performs the best possible WSD when $N = 10$. When $N < 5$, the performance of term sense disambiguation degrades; when $N > 10$, there is no obvious influence on the results. The default tuning of AP parameters is: $M = 1000, \lambda = 0.9$, and $convit = 50$. The custom tuning of these AP parameters produces no recognizable changes to the clustering results since the input similarities are always positive and in the range of $[0, 1]$. In *SemCluster*, the AP clustering preference is set to *median* to ensure that *SemCluster* performs clustering with higher granularity (i.e., a larger number of clusters) so that unimportant terms can be allocated to unimportant clusters and hence removed from the clustering results using ESA-based averaging.

As discussed in Sect. 3.2, τ, the distance threshold, has a direct impact on the performance of *SemCluster*. When $\tau = 1$, only the centroids of the clusters are chosen as seeds to identify and extract keyphrases; when $\tau = 0$, all the terms in T_D (except those belonging to the irrelevant clusters that have been discarded) are selected as seeds; hence most NP chunks in D are chosen as keyphrases. Considering the semantic similarity score of 0.5 as the least for which two terms can be judged semantically close, τ takes any value in the range $0.5 < \tau < 1$. Learning the optimal τ setting is a hyperparameter optimization problem that can be readily solved either by manual search or by utilizing optimization search algorithm [24]. In this experiment, we use manual search for τ estimation, and we randomly select 50 documents from Hulth's dataset to perform AKE using *SemCluster* with $\tau = 0.6, 0.7, 0.8$, and 0.9. *SemCluster* achieves the best possible performance on the selected samples when $\tau = 0.7$. The metric used for all *SemCluster* evaluations is `Precision/Recall/F1-measure`, which are defined as follows:

$$P = \frac{k_{correct}}{k_{extract}}, \qquad R = \frac{k_{correct}}{k_{standard}}, \qquad f = \frac{2 * PR}{P + R} \tag{6}$$

where $k_{correct}$ is the number of correct keyphrases extracted by *SemCluster*, $k_{extract}$ is the total number of the keyphrases extracted, and $k_{standard}$ is the total number of keyphrases assigned by human experts. Using the test partition of Hulth's dataset, *SemCluster* extracts 6974 keyphrases from 500 abstracts, among which 2737 phrases belong to the set of "gold-standard" keyphrases (i.e., $k_{standard}$). The results of the F1-measure and comparison with other methods are presented in Table 2.

Table 2. Comparison of AKE results for *SemCluster*, TextRank, and KeyCluster (SC/AP)

Method	Precision	Recall	F1-measure
TextRank	0.312	0.431	0.362
KeyCluster-SC	0.350	0.660	0.457
KeyCluster-AP	0.330	0.697	0.448
SemCluster	**0.392**	**0.721**	**0.507**

The results clearly show that *SemCluster* outperforms the other methods on both the recall and precision of keyphrases. Compared to KeyCluster-SC, the second best in performance, *SemCluster* achieves an improvement of 5% in F1-measure. Both *SemCluster* and KeyCluster-AP utilize the same clustering algorithm, but the former outperforms the latter with a 6% improvement in F1-measure. We observe that initial n-gram term selection and pruning unimportant clusters are the main contributors in boosting *SemCluster* precision. Likewise, the WordNet-based semantic representation of documents and expansion of seeds have a positive impact on keyphrase recall. According to the best of our knowledge, *SemCluster*'s F1-measure score of 0.507 using Hulth's dataset is the highest among current state-of-the-art unsupervised clustering-based methods.

SemCluster loads Wordnet directly into its memory (occupying about 22 MB) to give fast access to the semantics of any term in D. *SemCluster* accesses external knowledge bases only when a given term is not found in WordNet, which, compared with KeyCluster's crawling of about 50 million Wikipedia articles for every item to construct its conceptual vector, results in a significant improvement in performance.

5 Conclusion and Future Work

In this paper, we have introduced *SemCluster*, a clustering-based unsupervised keyphrase extraction method. By integrating an internal ontology (i.e., WordNet) with external knowledge sources, *SemCluster* identifies and extracts semantically important terms from a given document, clusters the extracted terms, identifies the most representive phrases and select among them the keyphrases via post-processing of the clustering results. The evaluation results verify the findings of Liu et al. [6] that unsupervised clustering-based AKE methods are effective and robust.

SemCluster is a part of a larger project specializing in big data processing called the Personal Data Lake (PDL) [15]. Though *SemCluster* is a general tool for extracting keyphrases from unstructured text, PDL is the main motivation behind it. PDL requires a domain-agnostic AKE tool to automatically process documents ingested from heterogenous data providers, and extract their keyphrases with the best possible precision and computational efficiency, and

thus generates semantic metadata for free-text data and allows them to interrelate on a micro-semantic level.

Although SemCluster exhibits good performance, there is still room for improvement. In an experiment on a collection of informal texts collected online from social networks and news websites, in which we replaced WuPalmer [19] with Jiang-Conrath [25] and used Babelfy [26] for term sense disambiguation, we observed an approximate 7% improvement in the F1-measure. However, computational efficiency decreased because Babelfy is an on line service. Nevertheless, this suggests a potential improvement to *SemCluster*, particularly by improving its semantic similarity measurement and word sense disambiguation.

Clustering, like all other clustering-based approaches, is at the heart of *SemCluster*. Improving the clustering performance improves the overall performance of *SemCluster*. Recently we became aware of a relatively new approach, an extension of Affinity Propagation, called AP clustering with seeds [27], which is reported to outperform the original AP algorithm. Testing this extension, or some modified version, on *SemCluster* over a variety of testing datasets is also one of our immediate undertakings in the future.

References

1. Turney, P.D.: Learning algorithms for keyphrase extraction. Inf. Retr. **2**(4), 303–336 (2000)
2. Hasan, K.S., Ng, V.: Automatic keyphrase extraction: a survey of the state of the art. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pp. 1262–1273 (2014)
3. Washio, T., Motoda, H.: State of the art of graph-based data mining. ACM SIGKDD Explor. Newsl. **5**(1), 59–68 (2003)
4. Sonowane, S.S., Kulkarni, P.A.: Graph based representation and analysis of text document: a survey of techniques. Int. J. Comput. Appl. **96**, 1–8 (2014)
5. Mihalcea, R., Tarau, P.: TextRank: bringing order into texts. Association for Computational Linguistics (2004)
6. Liu, Z., Li, P., Zheng, Y., Sun, M.: Clustering to find exemplar terms for keyphrase extraction. In: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pp. 257–266 (2009)
7. Steier, A.M., Belew, R.K.: Exporting phrases: a statistical analysis of topical language. In: Second Symposium on Document Analysis and Information Retrieval, pp. 179–190 (1993)
8. Barker, K., Cornacchia, N.: Using noun phrase heads to extract document keyphrases. In: Conference of the Canadian Society for Computational Studies of Intelligence, pp. 40–52. Springer (2000)
9. Litvak, M., Last, M.: Graph-based keyword extraction for single-document summarization. In: Proceedings of the Workshop on Multi-source Multilingual Information Extraction and Summarization, pp. 17–24 (2008)
10. Page, L., Brin, S., Motwani, R., Winograd, T.: The PageRank citation ranking: bringing order to the web. Technical Report, Stanford InfoLab (1999)
11. Tsatsaronis, G., Varlamis, I., Nrvg, K.: SemanticRank: ranking keywords and sentences using semantic graphs. In: Proceedings of the 23rd International Conference on Computational Linguistics, pp. 1074–1082 (1999)

12. Bracewell, D.B., Ren, F., Kuriowa, S.: Multilingual single document keyword extraction for information retrieval. In: Proceedings of 2005 IEEE International Conference on Natural Language Processing and Knowledge Engineering, pp. 517–522 (2005)
13. Hulth, A.: Improved automatic keyword extraction given more linguistic knowledge. In: Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing, pp. 216–223. Association for Computational Linguistics (2003)
14. Miller, G.A., Beckwith, R., Fellbaum, C., Gross, D., Miller, K.J.: Introduction to WordNet: an on-line lexical database. Int. J. Lexicogr. **3**(4), 235–244 (1990)
15. Alrehamy, H., Walker, C.: Personal data lake with data gravity pull. In: Proceedings of the 2015 IEEE Fifth International Conference on Big Data and Cloud Computing, pp. 160–167 (2015)
16. Navigli, R.: Word sense disambiguation: a survey. ACM Comput. Surv. **41**(2) (2009)
17. Patwardhan, S., Banerjee, S., Pedersen, T.: SenseRelate::TargetWord: a generalized framework for word sense disambiguation. In Proceedings of the ACL 2005 on Interactive Poster and Demonstration Sessions, pp. 73–76. Association for Computational Linguistics (2005)
18. Meng, L., Huang, R., Gu, J.: A review of semantic similarity measures in WordNet. Int. J. Hybrid Inf. Technol. **6**(1), 1–12 (2013)
19. Wu, Z., Palmer, M.: Verbs semantics and lexical selection. In: Proceedings of the 32nd Annual Meeting on Association for Computational Linguistics, pp. 133–138. Association for Computational Linguistics (1994)
20. Frey, B.J., Dueck, D.: Clustering by passing messages between data points. Science **315**(5814), 972–976 (2007)
21. Liu, F., Pennell, D., Liu, F., Liu, Y.: Unsupervised approaches for automatic keyword extraction using meeting transcripts. In: Proceedings of Human Language Technologies, pp. 620–628. Association for Computational Linguistics (2009)
22. Gabrilovich, E., Markovitch, S.: Computing semantic relatedness using Wikipedia-based explicit semantic analysis. In: Proceedings of the 20th International Joint Conference on Artificial Intelligence, pp. 1606–1611 (2007)
23. Witten, I., Milne, D.: An effective, low-cost measure of semantic relatedness obtained from Wikipedia links. In: Proceeding of AAAI Workshop on Wikipedia and Artificial Intelligence: An Evolving Synergy, pp. 25–30. AAAI Press, Chicago (2008)
24. Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. J. Mach. Learn. Res. **13**(2), 281–305 (2012)
25. Jiang, J.J., Conrath, D.W.: Semantic similarity based on corpus statistics and lexical taxonomy. arXiv preprint http://arxiv.org/abs/cmp-lg/9709008 (1997)
26. Moro, A., Cecconi, F., Navigli, R.: Multilingual word sense disambiguation and entity linking for everybody. In: Proceedings of the 2014 International Conference on Posters and Demonstrations, pp. 25–28. CEUR-WS.org (2014)
27. Guan, R., Shi, X., Marchese, M., Yang, C., Liang, Y.: Text clustering with seeds affinity propagation. IEEE Trans. Knowl. Data Eng. **23**(4), 627–637 (2011)