Measuring Term Informativeness in Context

Zhaohui Wu
Computer Science and Engineering
The Pennsylvania State University
University Park, PA 16802, USA
zzw109@psu.edu

C. Lee Giles
Information Sciences and Technology
Computer Science and Engineering
The Pennsylvania State University
University Park, PA 16802, USA
giles@ist.psu.edu

Abstract

Measuring term informativeness is a fundamental NLP task. Existing methods, mostly based on statistical information in corpora, do not actually measure informativeness of a term with regard to its semantic context. This paper proposes a new lightweight feature-free approach to encode term informativeness in context by leveraging web knowledge. Given a term and its context, we model context-aware term informativeness based on semantic similarity between the context and the term's most featured context in a knowledge base, Wikipedia. We apply our method to three applications: core term extraction from snippets (text segment), scientific keywords extraction (paper), and back-of-the-book index generation (book). The performance is state-of-the-art or close to it for each application, demonstrating its effectiveness and generality.

1 Introduction

Computationally measuring importance of a word in text, or "term informativeness" (Kireyev, 2009; Rennie and Jaakkola, 2005), is fundamental to many NLP tasks such as keyword extraction, text categorization, clustering, and summarization, etc. Various features derived from statistical and linguistic information can be helpful in encoding term informativeness, whereas practical feature definition and selection are usually ad hoc, data-driven and application dependent. Statistical information based on term frequency (TF) and document frequency (DF) tend to be more effective in finding keywords in large corpora, but can have issues with small amounts of text or small corpora. Linguistic information such as POS tag patterns often require manual selection based on prior applications. We contend that few methods actually measure the informativeness of a term to the discourse unit it contains. For example, given a context such as *"A graph comprises nodes (also called vertices) connected by links (also known as edges or arcs)"*, it is difficult to measure the term informativeness of "graph", "nodes", or "links" based on any statistical or linguistic information.

This raises many issues. Is there a fundamental and less ad hoc way to measure the term informativeness of a word within a discourse unit? Can we actually find a general approach based on comprehensive and high-level "knowledge" and not have to nitpick over features? Can this new metric be effectively applied to real world applications? To answer these questions, we develop a new term informativeness metric, motivated by query-document relevance in information retrieval. The higher the relevance score a query-document pair is, the more informative the query is to the document. If a similar principle also exists between word and context and there is an effective search engine returning ranked contexts for a given word, then we contend that word is more informative in the higher rank contexts. To see the term informativeness of three words "graph", "nodes" and "links" in context, we manually check the search results from Wikipedia, Google, and Bing. We found that very similar contexts are among the top 5 ranked results of "graph" while no such contexts appear in that of the other two words. Thus, we define a context-aware term informativeness based on the semantic relatedness

Proceedings of NAACL-HLT 2013, pages 259–269,
Atlanta, Georgia, 9–14 June 2013.

between the context and the term's featured contexts (or the top important contexts that cover most of a term's semantics).

We apply the context-aware term informativeness (CTI) to three typical NLP applications: core term extraction in snippets, keyword extraction and back-of-the-book index generation. Experiments show that the method is effective and efficient. Moreover, the metric can be easily combined with other methods, or as a feature for learning algorithms.

The remainder of this paper is organized as follows. Section 2 reviews the literature of term informativeness measurements. Section 3 proposes the formal definition of the context-aware term informativeness as well as its practical implementation using Web knowledge. Section 4 studies the three applications. Finally, we conclude with discussion and future work.

2 Related Work

Most known approaches to measure term informativeness fall into basically two categories: statistics-based and semantic-based.

Statistics-based methods, such as *TFIDF* (Salton and Buckley, 1988), *ResidualIDF*(*RIDF*), *Variance*, *Burstiness* and *Gain*, are based on derivations from term frequency (TF) and document frequency (DF). Sprck Jones defines *IDF* or *inverse document frequency* as:

$$IDF(w) = -log_2(df_w/D) \quad (1)$$

where D is the size of the corpus (Jones, 1972; Jones, 1973). Based on a finding that informative words tend to have large deviation between *IDF* and collection frequency f_w(the total number of occurrence of a word), many other informativeness scores have been proposed. Bookstein and Swanson (Bookstein and Swanson, 1974) introduced the x^I as:

$$X^I = f_w - df_w$$

Church and Gale (1995) introduced

$$variance(w) = \frac{1}{D-1}\sum_{d=1}^{D}(t_{dw} - \bar{t_w}) \quad (2)$$

where t_{dw} denotes w's TF in d and $\bar{t_w} = f_w/D$ indicates its mean expected word rate. Another measure suggested by them is

$$burstiness(w) = \frac{f_w}{df_w} \quad (3)$$

which tends to compare collection frequency and document frequency directly. Informative words were found to have *IDF* scores that are larger than what would be expected according to the Poisson model; *residual IDF* (*RIDF*) was introduced to measure this deviation

$$RIDF(w) = IDF(w) - \widehat{IDF}(w) \quad (4)$$

where $\widehat{IDF}(w) = -log_2(1 - e^{-\bar{t_w}})$. In addition, Papineni (2001) introduced the notion of *gain* as

$$gain(w) = \frac{df_w}{D}\left(\frac{df_w}{D} - 1 - log(\frac{df_w}{D})\right) \quad (5)$$

More recently, Rennie and Jaakkola (2005) introduced an informativeness score based on the fit of a word's frequency to a mixture of 2 Unigram distribution and applied it to named entity detection. It is worth noting that term necessity, which measures the probability that a term occurs in documents relevant to a given query, has been well studied in Information Retrieval community (Zhao and Callan, 2010; Yang and Callan, 2010). Though our CIT is not designed for probabilistic retrieval models, we may apply it to measure the term necessity in a query by considering it as a context.

Despite extensive research on semantic analysis and understanding of word and text (Deerwester et al., 1990; Budanitsky and Hirst, 2006; Cilibrasi and Vitanyi, 2007; Gabrilovich and Markovitch, 2007; Agirre et al., 2009; Yazdani and Popescu-Belis, 2012), little work studied the measurement of the semantics of term informativeness. An exception is the $LSAspec$ from Kireyev (2009), based on latent semantic analysis (Deerwester et al., 1990), which is defined as the ratio of a term's LSA vector length to its document frequency and thus can be interpreted as the rate of vector length growth. However, latent semantic models such as LSA are notoriously hard to interpret since the "latent concepts" cannot be readily mapped to human knowledge (Gabrilovich and Markovitch, 2007). Our approach explicitly leverages the semantics of word and text using existing knowledge bases.

Previous methods, all corpus-based, might be effective in identifying informative words at the document or corpus level, but do not the ability to capture term informativeness in a particular context due to their absence of semantics and obliviousness of context. Our method measures the term informativeness within a context in a semantic-based approach, regardless of the absence of statistical information.

3 Context-aware Term Informativeness

3.1 Context

A context of a word or phrase may refer to a few words nearby (He et al., 2010), a sentence or paragraph (Soricut and Marcu, 2003), or even a set of documents containing it (Cilibrasi and Vitanyi, 2007). Here we define context as a syntactic unit of discourse such as a sentence or paragraph, for example, "PL/SQL is one of three key programming languages embedded in the Oracle Database", or "There are two types of functions in PL/SQL". The universal context set $U(t)$ of a word t is defined as all the contexts containing it in the web. Different contexts vary in their authority just like web pages vary. For the two examples, we could argue that the first context is much more "authoritative" than the second. This can be verified by their popularity on Google; (all results from actual search engines were at the time of this publication) the first retrieves approximately 302,000 exact matching results while the second retrieves only one. We consider this as the number of citations of a context, which, to some extent, indicates its "authority". We define the *source* of a context as the set of all documents citing it. Here "citing" instead of "containing" is used because some documents may not literally contain an exact copy of the context.

Given a term t, define its universal context set $U(t) = \{c_i\}$ and the source of c_i is $S(c_i) = \{d_{ij}\}$. Ideally, the authority of a context will be contributed by every document citing it. Therefore, we define the authority score of a context as

$$CA(c_i) = \sum_j DA(d_{ij}) \tag{6}$$

where $DA(d_{ij})$ denotes the authority contributed by d_{ij}. It is very difficult to acquire the universal context set of a term. Considering that usually we only care about the top few results of a query returned by search engines and ignore a large faction of less important ones, it is reasonable to assume that a term's semantics will be well covered by a few important contexts. We therefore define the featured context set of term t, or $U_f(t)$, as the top k contexts with the highest authority scores, where k is an application dependent parameter. In our experiments, the default k for the Wikipedia based implementation is 20.

3.2 Term Informativeness

We now consider how to measure the term informativeness in context. Using the context "PL/SQL is one of three key programming languages embedded in the Oracle Database" (denoted by C_p) as an example, for its term "PL/SQL", the top three contexts returned by Google are

1. PL/SQL (Procedural Language/Structured Query Language) is Oracle Corporation's procedural extension language for SQL and the Oracle relational database.
2. PL/SQL is Oracle's procedural extension to industry-standard SQL. PL/SQL naturally, efficiently, and safely extends SQL.
3. This Oracle PL SQL tutorial teaches you the basics of programming in PL/SQL like cursors, stored procedures, PlSQL functions.

Those contexts, though being diverse in actual meaning, all have semantic relatedness to C_p. Even someone who does not completely understand them can gain some meaning by observing common words such as "Oracle", "database" and "programming". However, checking the Google results for "Oracle Database" or "programming languages", we will find little relatedness between them and C_p. This suggests that if term t_a in context c is more informative than t_b, then most likely the contexts from t_a's featured context set will be more related to c than will t_b. Thus, given a term t and its featured context set $U_f(t) = \{c_1, ..., c_k\}$, we define the term informativeness of t in context c_i as

$$I(t, c_i) = \sum_{c_j \in U_f(t)} \kappa(c_i, c_j) \cdot CA(c_j) \tag{7}$$

where $\kappa(c_i, c_j)$ is the semantic relatedness of c_i and c_j, which can be computed by various semantic relatedness metrics such as Wikipedia based (Gabrilovich and Markovitch, 2007; Yazdani

and Popescu-Belis, 2012), Wordnet based (Agirre et al., 2009; Budanitsky and Hirst, 2006), or simple cosine similarity and Jaccard similarity based (Zobel and Moffat, 1998).

The context-aware term informativeness (CTI) introduced above is a formal and general definition. As such the definition in Equation (7) includes several features such as context authority score, featured context set, semantic relatedness, and knowledge base, any or all of which could be flexible for different applications.

3.3 Implementation

Here, we present a simple practical implementation using Wikipedia as the knowledge base and the context authority estimated by the discounted rank of the Wikipedia document. Note that the problem is how to compute $CA(c_j)$ for each context in U_f. We rewrite Equation (6) as

$$CA(c_i) = DA(d_{i0}) + \sum_{j \neq 0} DA(d_{ij}) \qquad (8)$$

where d_{i0} is the original document of c_i and all the others are further derivatives of "citing" c_i. For example, the Wikipedia page of "PL/SQL" will be considered as the original document of C_p while all other documents citing C_p are its derivatives. Intuitively, the authority of a context will mainly rely on the authority of its original document. Here, we simply assume that the context authority depends only on its original document, or

$$CA(c_i) \approx DA(d_{i0}) \qquad (9)$$

We then take the top ranked document returned by the web knowledge base as the original document. We present a practical implementation of CTI in Algorithm 1. The discounted rank is used to represent the relative context authority score of each context in U_f. We use Wikipedia as our knowledge base to implement the metric since it is currently one of the largest and most readily available knowledge repositories and, more importantly, provides free, unlimited and fast query APIs[1]. Given any keyword, the Wikipedia query API will return the ranked Wikipedia entries along with the contexts containing the keyword. We set the default value 20 for k, or $len(U_f)$. Note that there could be other variations of this implementation. For example, we could rule out duplicate or very similar results in the U_f. Search engines such as Google and Bing are also potential sources since they return high quality web pages along with the contexts containing the query keyword.

In terms of scalability, the proposed method is inherently parallelizable, not only at the document level, but also a the context level, since computing CTI does not depend on any other context in the document. In addition, we do not need to issue the same query more than once. Our strategy is to locally cache the returned results of every seen query. For a new term seen in a previous query, we can directly access the local cached file. If we have built a large local pool, the queries will rarely go to a search engine or other source. Given a corpus size N (words in total), the number of actual issued queries will be at most the number of unique terms, which is far less than O(N). Of course, new terms never seen will have to be processed, but there will be fewer of these over time.

Algorithm 1: Wikipedia-based $I(t, c_i)$

```
Input: t, c_i
Output: I(t, c_i)
begin
    I ⟵ 0;
    U_f ⟵ queryWikipedia(t);
    for j ∈ range(len(U_f)) do
        s ⟵ κ(c_i, U_f[j]);
        if j > 0 then
            I ⟵ I + s/log(j + 1);
        else I ⟵ I + s
    return I;
```

4 Applications

4.1 Core Terms Extraction from Snippets

We first investigate CTI in a well defined setting. That is, if we have a collection of terms such that its most important context is a "definition," e.g. "*database*" and "*A database is a structured collection of data, which are typically organized to model relevant aspects of reality, in a way that supports*

[1] http://www.mediawiki.org/wiki/API:Query

Exemplary snippets of computer science terms	Top 5 terms ranked by CTI
Acrobat, a document exchange software from Adobe Systems, provides a platform-independent means of creating, viewing, and printing documents. Acrobat can convert a DOS, Windows, UNIX or Macintosh documents into a Portable Document Format (PDF) which can be displayed on any computer with an Acrobat reader. The Acrobat reader can be downloaded free from the Adobe website.	Acrobat:3.19 Acrobat reader:2.94 Portable Document Format:2.08 Adobe website:2.03 Adobe Systems:1.82
Data mining (DM), also known as Knowledge-Discovery in Databases (KDD) or Knowledge-Discovery and Data Mining (KDD), is the process of automatically searching large volumes of data for patterns. Data mining uses automated data analysis techniques to uncover previously undetected relationships among data items. Data mining often involves the analysis of data stored in a data warehouse. Three of the major data mining techniques are regression, classification and clustering.	data mining:3.77 data mining techniques:3.64 KDD:1.79 Knowledge-Discovery:1.66 data analysis techniques:1.20
Firefox, also known as Mozilla Firefox, is a free, open source, cross-platform, graphical web browser developed by the Mozilla Corporation and hundreds of volunteers. Firefox includes an integrated pop-up blocker, tabbed browsing, live bookmarks, support for open standards, and an extension mechanism for adding functionality. Although other browsers have some of these features, Firefox became the first such browser to include them all and achieve wide adoption.	Mozilla Firefox:3.89 firefox:3.13 web browser:2.44 browser:2.39 graphical web browser:2.35

Table 1: Term ranked by CTI from exemplary snippets

processes requiring this information", can CTI identify "database" as the most informative term in this context? To construct the term-context pairs, we could use the Wikipedia title and the top ranked context returned by searching the title using the Wikipedia API. Then we could test our metric based on other search engines such as Google or Bing. Testing manually, we found the results compare well to the search engine results, since both Google and Bing give top ranks to Wikipedia pages if the query keyword is a Wikipedia title. For further analysis, we need a collection of term-context pairs from other sources different from Wikipedia. Fortunately, we found a list of 1255 computer science terms with its definition snippets manually created by Web users [2]. The snippets are literally different from those contexts in Wikipedia and some of the terms are even not Wikipedia titles, e.g. bBlog, BetBug, etc. These can be part of an "initial" evaluation. The core term extraction algorithm works in the following steps for each term-context pair:

1. Extract all n-grams ($1 \leq n \leq 4$) in the context as candidates
2. For each candidate, calculate its CTI using Wikipedia based implementation
3. Return the top K highest CTI as core terms

We used the top 20 returned Wikipedia contexts as a featured context set U_f and apply the cosine similarity for κ. We show some exemplary snippets with its top 5 core terms and their CTI scores in Table 1. The overall performance is shown in Table 2, in terms of precision, recall and F1 scores based on the only one titled term of each snippet as the ground truth. CTI can correctly find the core term for 37.5% snippets. If we take the top 5 results, then the recall increase to 76.3%.

[2]http://www.labautopedia.org/mw/index.php/List_of_programming_and_computer_science_terms

K	Precision (%)	Recall (%)	F1(%)
1	37.5	37.5	37.5
2	35.1	55.2	42.9
3	32.3	64.7	43.1
4	31.3	72.2	43.7
5	27.6	76.3	40.5
10	20.0	88.1	32.6

Table 2: Results on computer science term extraction from descriptive snippets

Though the algorithm can be easily parallelized, sequentially runtime on all snippets took only slightly more than a minute on a 2.35GHz Intel(R) Xeon(R) 4 processors, 23GB of RAM, and Red Hat Enterprise Linux Server(5.7) machine. However, the time could vary due to network conditions.

Though these results look promising, but it could be due to the high lexical similarity between this dataset and Wikipedia content. To test on a more general corpora, we explore more real world tasks.

4.2 Keyword Extraction

There is a rich literature on keyword extraction problem (Frank et al., 1999; Witten et al., 1999; Turney, 2000; Hulth et al., 2003; Tomokiyo and Hurst, 2003;

	Wiki20			citeulike180		
Method	P	R	F	P	R	F
TFIDF	13.7	17.8	15.5	14.4	16.0	15.2
KEA	18.4	21.5	19.8	20.4	22.3	21.3
CTI	19.6	22.7	21.0	18.5	21.4	19.8

Table 3: Results on Wiki20 and citeulike180

Method	Precision (%)	Recall (%)	F1(%)
TFIDF	14.9	15.3	15.1
HUMB	27.2	27.8	27.5
CTI	19.3	20.1	19.7
CTI+	25.3	26.2	25.7

Table 4: Results on SemEval2010

Mihalcea and Tarau, 2004; Medelyan and Witten, 2008; Liu, 2010), most of which is treated as a classification or ranking problem with corresponding machine learning algorithms that use statistical and linguistic features in a corpus. Here, we consider the task as finding the most informative keywords in a document. Given a document $d = \{c_i\}$, our keyword extraction algorithm based on CTI works as follows.

1. For each context c_i in a document, compute the semantic relatedness $s(c_i, d)$ between c_i and d
2. For each n-gram ($1 \leq n \leq 4$) t in c_i, calculate $I(t, c_i)$ using Wikipedia based implementation
3. Select the top keywords with the highest $\sum_i I(t, c_i) * s(c_i, d)$

Note that for the last step keywords are selected based on a summarized weighted informativeness score over a document. Obviously, the pure cosine or Jaccard similarity is not a good choice to measure semantic relatedness between two text segments of very low lexical similarity. We thus use the Wikipedia based ESA (Gabrilovich and Markovitch, 2007) to compute the semantic relatedness $s(c_i, d)$ and $\kappa(c_i, c_j)$. To make the calculation more efficient, only the Wikipedia pages whose title is contained in the dataset are used to build the concept space. We ran the algorithm on several datasets including Wiki20 (Medelyan et al., 2008), citeulike180 (Medelyan et al., 2009) and SemEval2010 (Kim et al., 2010) [3].

Though keyword extraction as a research topic has a rich literature, to the best of our knowledge there is no large scale datasets publicly available. The Wiki20 dataset contains 20 computer science articles each with around 5 terms labeled by 15 different teams. Every term is a Wikipedia title. The citeulike180 contains a set of 180 papers each tagged with around three tags by 332 users. For each dataset, the collection of all labeled keywords by different taggers are considered as the gold standard for a document. We use the set of all keywords for evaluation; otherwise a more complicated evaluation metrics for each dataset will be needed. It would be better to investigate other weighting schemes. However, the datasets here are relatively small and the number of tags on which at least two annotators agreed is significantly small; weighting the keywords might not make too much difference. KEA [4] builds a Naive Bayes model using features TFIDF, first occurrence, length of a phrase, and node degree (number of candidates that are semantically related to this phrase) (Witten et al., 1999). First occurrence is computed as the percentage of the document preceding the first occurrence of the term in the document. We compute the node degree as the textrank (Mihalcea and Tarau, 2004) degree in a document by simply relating two candidate terms with each other if they are in the same context. KEA uses 5 fold cross validation. All precision P, recall R and F1 F results are over the top 10 candidate keywords and the micro-averaged results of the first two datasets are shown in Table 3. The CTI-based algorithm works better than KEA on Wiki20 but slightly worse on citeulike180. We argue that the reason might be two-fold. First, CTI does not use any inter-document or corpus information while KEA learns from the corpus. As such, CTI might not perform as well as supervised learning methods for a domain dependent large corpus. Second, the labeled keywords in Wiki20 are all Wikipedia titles while those in citeulike are general tags labeled by voluntary web users. CTI would give more preference to Wikipedia titles since their featured context set returned from Wikipedia is more semantically representative than other non-Wikipedia title words.

[3] http://code.google.com/p/maui-indexer/downloads/list

[4] http://www.nzdl.org/Kea/

Dataset	#Books	#Words	#Contexts	Main domains
Gutenberg	55	7,164,463	301,581	History, Art, Psychology, Philosophy, Literature, Zoology
Open Book	213	22,279,530	1,135,919	Computer Science, Engineering, Information Science

Table 5: Datasets for book index generation evaluation

The SemEval2010 dataset contains a set of 284 scientific papers with 15 keyphrases assigned by readers and authors. 144 of them are selected as training set while the other 100 are for testing. A comparison of CTI to the results from TFIDF and the best reported results HUMB (Lopez and Romary, 2010) is shown in Table 4. It achieves 19.8% by micro-averaged F1 score, ranking 11th out of the 19 systems submitted to the competition (Kim et al., 2010). However, by adding the structural features used by HUMB into CTI, we can improve the performance by around 6%, making our results close to that of HUMB. The structural information is encoded as weights for context that is located in title, abstract, section titles and general content. Each weight can be regarded as the prior probability that a keyword will appear in the corresponding location, whose value can be set according to the fraction of the number of keyword occurrences of this type of location with respect to the number of all keyword occurrences in the entire training set. Here they are set to be 0.3, 0.4, 0.25, and 0.05.

4.3 Back-of-the-book Index Generation

A back-of-the-book index (or book index) is a collection of words or phrases, often alphabetically arranged as an index, created to give readers important location of important information in a given book. Usually indexing is done by freelancers hired by authors or publishers, namely professional indexers [5]. Csomai and Mihalcea first evaluated the performance of different informativeness measurements for selecting book index terms (2007) and then investigated automatic book index generation in a supervised learning framework (2008) using syntactic features, linguistical features, encyclopedic features, etc., as a keyword extraction problem rather than building a actual book index.

A set of keywords is not a back-of-the-book index. What really matters for such an index is that an index term or phrase points to its proper location in the text. For example, in "pattern recognition and machine learning" by Bishop, "hidden Markov model" appears in more than 20 pages while the actual index entry has only 2 pages as its locators. Thus the actual problem is to identify a index term with its context. As such, learning a robust and efficient model for real book indexes is challenging. First, books from different domains vary in vocabulary composition and structure style, requiring various indexing specialties. There are different indexing guides for medicine (Wyman, 1999), psychology (Hornyak, 2002), and law (Kendrick and Zafran, 2001). Second, book indexing is a highly subjective work and indexes of different books are always created by different professional indexers who have their own preferences and background (Diodato and Gandt, 1991; Diodato, 1994). Third, the training set is extremely unbalanced. As we found in our dataset, the index size is only 0.42% of the length of book on average. All these motivate us to explore the automatic creation of index terms that are aware of the context at the term's locations (locators). To do so we propose the following efficient training-free and domain independent approach:

1. For each context c_i in a book, compute its weight w_i based on structural features
2. For each candidate term t in c_i, calculate $I(t, c_i)$ using Wikipedia based implementation
3. Select term-context pairs with the highest $w_i * I(t, c_i)$ as index entries

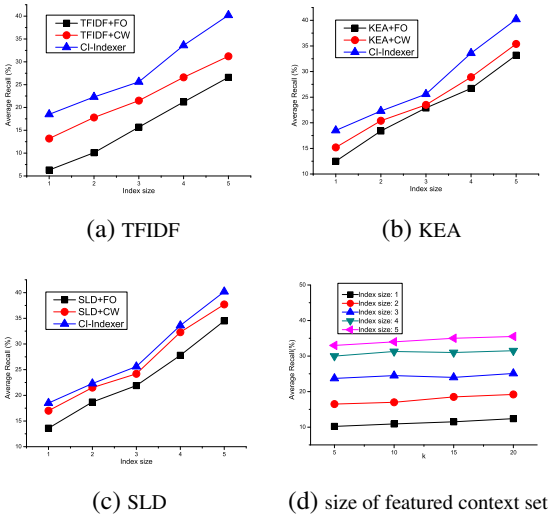
The weight in step 1 represents the relative importance of a context in a book. $w(c) = 1 - \frac{c_{id}(c) - c_{id}(title_c)}{N_{title_c}}$ measures the weight based on the normalized distance from the context to its direct chapter or sub-chapter title, where $c_{id}(c)$ denotes the id of context c, $title_c$ the title of context c and N_{title_c} the number of contexts under $title_c$. To select candidate terms, we first filter the improbable index terms

[5] http://www.asindexing.org/

based on POS patterns using the Standard POS Tagger (Toutanova et al., 2003). We then select multiword keyphrases based on Pointwise Mutual Information (PMI) (Church and Hanks, 1990), which was shown to be the best metric to measure word associations (Terra and Clarke, 2003).

To evaluate our back-of-the-book index generation method, we conduct extensive experiments on books in various domains, from the Gutenberg dataset and the open book dataset described in Table 5. The first one was created by (Csomai and Mihalcea, 2006), containing 55 free books collected from Gutenburg[6]. Since the dataset does not provide the locators of index terms, we can only serve the evaluation as a keyword extraction task. The second dataset was collected from CiteSeer repository, most of which are in computer science and engineering. We extracted the paged body text and the back index using Pdfbox[7]. Having each index term associated with its locators (page numbers), we can perform an evaluation for different methods, not based solely on keyword extraction.

We first compare CTI with other metrics on both datasets for keywords extraction since all other metrics are context-oblivious. CTI selects index terms based on the sum of a term's CTI scores over all its contexts, the same as the algorithm used in Section 4.2. The results are shown in Table 6, where the index size = n indicates the number of output terms is n times of the true book index size for each book. The scores are the average recall over a dataset. The CTI outperforms all other 7 metrics in the two datasets as the output index size increases. Moreover, results show that TF and TFIDF are better than RIDF in identifying book index terms, which seems contradictory to previous findings (Church and Gale, 1995). A possible reason is that a book is much longer than a regular document thus enhancing TF as a better indicator of keywords but weakening the role of IDF. We believe this is why *Variance*, *Gain*, and *Burstiness*, which relies on DF, are less effective here. *Wikipedia keyphraseness* (Csomai and Mihalcea, 2008) can only find a small fraction of index terms because it emphasizes Wikipedia titles that have high in-degree in hyper-link network formed by Wikipedia terms. However, a book index covers much broader terms not titled in Wikipedia.



(a) TFIDF (b) KEA

(c) SLD (d) size of featured context set

Figure 1: Results for book index generation

We then compare with three baselines TFIDF, KEA, and SLD (supervised learning using decision tree in Csomai's (2008)) on the second dataset. For SLD, we use all the features except the discourse comprehension based ones which were too complicate to implement. We choose a decision tree because its training is much faster than the other two models while its performance is quite close to the best. We follow Csomai's setting to choose 90%(192) books for training and the other 10%(21) for test. We set two strategies to make the baselines context-aware. First, we select the page of a term's first occurrence as its locator, denoted by "+FO" in Figure 1. Second, we apply the context weight to them, denoted by "+CW". "CI-Indexer" denotes our method. The results are shown in Figure 1a, 1b and 1c respectively. For all the three baselines, adding context weight gives better performance than using the simple first occurrence guess, especially for TFIDF. KEA benefits least from the context weights, suggesting its first occurrence and node degree features play a similar role as the context weight features. SLD outperforms TFIDF and KEA under both strategies probably because of the new features of POS pattern and Wikipedia keyphraseness. "SLD+CW" is the closest to ours. Finally, we show in Figure 1d that increasing the size of featured context set for CTI from 5 to 20 can slightly improve

[6]www.gutenberg.org/
[7]pdfbox.apache.org/

Dataset	Open book dataset					Gutenberg dataset				
Index size	1	2	3	4	5	1	2	3	4	5
Variance	2.4	4.8	7.5	10.4	13.4	1.1	2.9	5.3	8.0	11.0
Gain	2.9	6.4	10.2	14.3	18.2	4.9	9.0	14	18.6	23.0
Wikipedia keyphraseness	5.3	9.5	13.5	16.4	20.5	9.2	14.1	18.5	21.4	24.3
Burstiness	6.0	11.4	16.6	21.4	25.8	10.0	15.8	20.2	23.1	26.2
RIDF	8.6	14.5	19.5	23.9	28.0	10.4	15.9	20.1	23.2	26.3
TF	9.8	16.9	23.3	29.0	31.7	10.4	17.6	23.5	28.1	30.7
TFIDF	10.3	17.3	23.8	29.3	33.6	11.8	19.9	24.7	28.9	32.9
CTI	**12.4**	**19.2**	**25.1**	**31.5**	**35.5**	**14.9**	**22.3**	**26.9**	**29.3**	**34.5**

Table 6: Average recall(%) comparisons as the output index size increases

performance in different index size settings.

4.4 Discussion

The three applications are (incrementally) designed for different goals. The first is a toy application to show the potential capability of this approach, regardless of syntactic or statistical information. Clearly, there are simple heuristics that can work very well for this task, e.g. the first term of the context. TF or TFIDF also performs quite well. We can rewrite each context (by reordering the terms, changing sentence structures, or substituting the core terms with pronouns) to make them ineffective. However, this will not effect our method, because what it essentially measures is a term's informativeness among a list of terms appearing in the same context. However, for keyword extraction, a topic with a rich literature, to the best of our knowledge, has no publicly available large scale datasets, which makes SemEval2010 the best available. We believe our application on back-of-the-book index generation showed how CTI can scale real world large corpora and will scale to millions of books since each book can be processed separately.

Based on the applications we explored, we can see that the practical utility of CTI used alone could be limited, especially for context-oblivious tasks. It seems reasonable that this method does not outperform supervised learning methods designed for keyword extraction. However, our method shows what simple but elegant methods can achieve without the overhead of machine learning, especially for context-aware scenarios such as finding book index terms.

5 Conclusion and Future Work

We developed a new web knowledge based method for encoding informativeness of terms within a unit of discourse. It is totally feature-free, corpus-free, easy to implement, and inherently parallelizable. Three typical applications on text snippets, scientific papers and non-fiction books show its effectiveness. The segmentation of context, the size of featured context set, the semantic relatedness metric κ, and the knowledge base might more or less affect the final performance of CTI in terms of accuracy or efficiency. For all applications, we treat a paragraph as an individual context, which is not necessary a complete discourse unit. However, it may not be fair to set the same number for all context terms. In addition, selection of semantic relatedness and knowledge bases need further investigation. The Wikipedia-based implementation might be a good choice for the definitional snippets, scientific articles and text books since they are all "educational" resources sharing a similar concept space. However, it is an open question as whether it works for corpora such as tweets, online reviews, and forum posts.

Based on the proposed methods and encouraging results, it would be interesting to build an online indexing tool which automatically finds informative terms in generic text and generates a back-of-the-book index for a sets of papers, books, theses and other collections.

6 Acknowledgments

We gratefully acknowledge partial support from the National Science Foundation and useful comments from referees.

References

E. Agirre, E. Alfonseca, K. Hall, J. Kravalova, M. Pasca, and A. Soroa. 2009. A study on similarity and relatedness using distributional and WordNet-based approaches. In *Proceedings of NAACL-HLT* 2009, pp. 19–27.

A. Bookstein and Don R. Swanson. 1974. Probabilistic models for automatic indexing. *Journal of the American Society for Information Science*, 25(5):312–316.

A. Budanitsky and G. Hirst. 2012. Evaluating WordNet-based measures of semantic distance. *Computational Linguistics* 32:13–47.

K. W. Church and W. A. Gale. 1995. Inverse document frequency(IDF): A measure of deviation from poisson. In *Proceedings of the Third Workshop on Very Large Corpora* 1995, pp. 121–130.

K. W. Church and P. Hanks. 1990. Word association norms, mutual information and lexicography. *Computational Linguistics*, 16:22–29.

Rudi L. Cilibrasi and Paul M.B. Vitanyi. 2007. The Google similarity distance. *IEEE Transaction on Knowledge and Data Engineering*, 19(3):370–383.

A. Csomai and R. Mihalcea. 2006. Creating a testbed for the evaluation of automatically generated back-of-the-book indexes. In *Proceedings of the 7th international conference on Computational Linguistics and Intelligent Text Processing* 2006, pp. 429–440.

A. Csomai and R. Mihalcea. 2007. Investigations in unsupervised back-of-the-book indexing. In *Proceedings of the Florida Artificial Intelligence Research Society* 2007, pp. 211–216.

A. Csomai and R. Mihalcea. 2008. Linguistically Motivated Features for Enhanced Back-of-the-Book Indexing. In *Proceedings of ACL* 2008, pp. 932–940.

S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. 1990. Indexing by Latent Semantic Analysis. *Journal of the American Society for Information Science* 41:391-407.

V. Diodato and G. Gandt. 1991. Back of book indexes and the characteristics of author and nonauthor indexing: Report of an exploratory study. *Journal of the American Society for Information Science*, 42:341–350.

V. Diodato. 1994. User preferences for features in back of book indexes. *Journal of the American Society for Information Science*, 45:529–536.

E. Frank, G. W. Paynter, I. H. Witten, and C. Gutwin. 1999 Domainspecic keyphrase extraction. In *Proceedings IJCAI* 1999, pp. 668-673.

E. Gabrilovich and S. Markovitch. 2007. Computing semantic relatedness using Wikipedia-based explicit semantic analysis. In *Proceedings of IJCAI* 2007, pp. 6–12.

Q. He, J. Pei, D. Kifer, P. Mitra, and C. L. Giles. 2010. Context-aware citation recommendation. In *Proceedings of WWW* 2010, pp. 421–430.

B. Hornyak. 2002. Indexing Specialties: Psychology. Medford, NJ : Information Today, Inc.

A. Hulth. 2003. Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of EMNLP* 2003, pp. 216-223.

Karen Sprck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28(1):11–21.

Karen Sprck Jones. 1973. Index term weighting. *Information Storage and Retrieval*, 9(11):619–633.

P. Kendrick and E. L. Zafran. 2001. Indexing Specialties: Law. Medford, NJ : Information Today, Inc.

Su Nam Kim, Olena Medelyan, Min-Yen Kan, and Timothy Baldwin. 2010. Semeval-2010 task 5 : Automatic keyphrase extraction from scientic articles. In *Proceedings of the 5th SIGLEX Workshop on Semantic Evaluation.* 2010, pp. 21–26.

K. Kireyev. 2009. Semantic-based Estimation of Term Informativeness. In *Proceedings of NAACL-HLT* 2009, pp. 530–538.

Z. Liu, W. Huang, Y. Zheng, and M. Sun. 2010. Automatic keyphrase extraction via topic decomposition. In *Proceedings of EMNLP* 2010, pp. 366–376.

P. Lopez and L. Romary. 2010. HUMB: Automatic Key Term Extraction from Scientic Articles in GROBID. In *Proceedings of the 5th SIGLEX Workshop on Semantic Evaluation.* 2010, pp. 248-251.

O. Medelyan and I. H. Witten. 2008. Domain-independent automatic keyphrase indexing with small training sets. *J. Am. Soc. Inf. Sci. Technol.* 59:1026-1040.

O. Medelyan, I. H. Witten, and D. Milne. 2008. Topic indexing with Wikipedia. In *Proceedings of AAAI08 Workshop on Wikipedia and Artificial Intelligence: an Evolving Synergy* 2008, pp. 19–24.

O. Medelyan, E. Frank, and I. H. Witten. 2009. Human-competitive tagging using automatic keyphrase extraction. In *Proceedings of EMNLP* 2009, pp. 1318–1327.

R. Mihalcea and P. Tarau. 2004. TextRank: Bringing Order into Texts. In *Proceedings of EMNLP* 2004, pp. 404-411.

L. Page, S. Brin, R. Motwani, and T. Winograd. 1998. The PageRank Citation Ranking: Bringing Order to the Web. *Technical Report.* Stanford InfoLab.

K. Papineni. 2001. Why inverse document frequency? In *Proceedings of NAACL-HLT* 2001, pp. 1–8.

J. D. M. Rennie, and T. Jaakkola. 2005. Using Term Informativeness for Named Entity Detection. In *Proceedings of SIGIR* 2005, pp. 353–360.

G. Salton, and C. Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information Processing & Management* 24(5):513–523.

R. Soricut and D. Marcu. 2003. Sentence level discourse parsing using syntactic and lexical information. In *Proceedings of NAACL-HLT* 2003, pp. 149–156.

E. Terra and C. L. Clarke. 2003. Frequency estimates for statistical word similarity measures. In *Proceedings of NAACL-HLT* 2003, pp. 165–172.

T. Tomokiyo and M. Hurst. 2003. A language model approach to keyphrase extraction. In *Proceedings of the ACL 2003 workshop on Multiword expressions: analysis, acquisition and treatment*, 2003, pp. 3340.

K. Toutanova, D. Klein, C. Manning, and Y. Singer. 2003. Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network. In *Proceedings of NAACL-HLT* 2003, pp. 252-259.

P. D. Turney. 2000. Learning Algorithms for Keyphrae Extraction. *Information Retrieval* 2:303-336.

I. H. Witten, G. W. Paynter, E. Frank, C. Gutwin, and C. G. Nevill-Manning. 1999. Kea: practical automatic keyphrase extraction. In *Proceedings of the fourth ACM conference on Digital libraries*, 1999, pp. 254–255.

L. P. Wyman. 1999. Indexing Specialities: Medicine. Medford, NJ : Information Today, Inc.

M. Yazdani and A. Popescu-Belis. 2012. Computing text semantic relatedness using the contents and links of a hypertext encyclopedia. *Artificial Intelligence* 194:176–202.

J. Zobel and A. Moffat. 1998. Exploring the similarity space. *ACM SIGIR Forum* 32(1):18–34.

Le Zhao and Jamie Callan. 2010. Term necessity prediction. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, 2010, pp. 259–268.

Hui Yang and Jamie Callan. 2009. A metric-based framework for automatic taxonomy induction. In *Proceedings of the ACL*, 2009, pp. 271–279 .