IDF for Word N-grams

MASUMI SHIRAKAWA, TAKAHIRO HARA, and SHOJIRO NISHIO, Osaka University

Inverse Document Frequency (IDF) is widely accepted term weighting scheme whose robustness is supported by many theoretical justifications. However, applying IDF to word N-grams (or simply N-grams) of any length without relying on heuristics has remained a challenging issue. This article describes a theoretical extension of IDF to handle N-grams. First, we elucidate the theoretical relationship between IDF and information distance, a universal metric defined by the Kolmogorov complexity. Based on our understanding of this relationship, we propose N-gram IDF, a new IDF family that gives fair weights to words and phrases of any length. Based only on the magnitude relation of N-gram IDF weights, dominant N-grams among overlapping N-grams can be determined. We also propose an efficient method to compute the N-gram IDF weights of all N-grams by leveraging the enhanced suffix array and wavelet tree. Because the exact computation of N-gram IDF provably requires significant computational cost, we modify it to a fast approximation method that can estimate weight errors analytically and maintain application-level performance. Empirical evaluations with unsupervised/supervised key term extraction and web search query segmentation with various experimental settings demonstrate the robustness and language-independent nature of the proposed N-gram IDF.

CCS Concepts: • **Mathematics of computing** → **Information theory**; • **Information systems** → **Content analysis and feature selection**; • **Theory of computation** → **Data structures design and analysis**; • **Computing methodologies** → *Natural language processing;*

Additional Key Words and Phrases: Term weighting, multiword expression, Kolmogorov complexity, information distance, wavelet tree, Poisson distribution

ACM Reference Format:
Masumi Shirakawa, Takahiro Hara, and Shojiro Nishio. 2017. IDF for word N-grams. ACM Trans. Inf. Syst. 36, 1, Article 5 (June 2017), 38 pages.
DOI: http://dx.doi.org/10.1145/3052775

1. INTRODUCTION

Term weighting schemes as represented by Term Frequency-Inverse Document Frequency (TF-IDF) [Salton et al. 1975] are fundamental technologies for text analysis. TF-IDF was originally introduced as a weighting factor for each word in document retrieval where a document is represented by a vector of words that occur in the given

This article is an extended version of a paper presented at the 24th International World Wide Web Conference [Shirakawa et al. 2015].

This research is partially supported by a Grant-in-Aid for Scientific Research (A)(2620013) from the Ministry of Education, Culture, Sports, Science and Technology (MEXT), Japan; JST, Strategic International Collaborative Research Program, SICORP; the CPS-IIP Project (Integrated Platforms for Cyber-Physical Systems to Accelerate Implementation of Efficient Social Systems) in the research promotion program for national level challenges "research and development for the realization of next-generation IT platforms" by MEXT, Japan.

Authors' addresses: M. Shirakawa and T. Hara, Department of Multimedia Engineering, Graduate School of Information Science and Technology, Osaka University, 1-5 Yamadaoka, Suita, Osaka 565-0871, Japan; emails: {shirakawa.masumi, hara}@ist.osaka-u.ac.jp; S. Nishio, Osaka University, 1-1 Yamadaoka, Suita, Osaka 565-0871, Japan; email: nishio@ist.osaka-u.ac.jp.

DOI: http://dx.doi.org/10.1145/3052775

document. TF-IDF has become a de facto standard term weighting scheme for bag-of-words representation of texts in information retrieval and text mining. Recently, it has been used to explicitly highlight key terms in texts in natural language processing (NLP) [Hasan and Ng 2010].

Term weighting schemes can usually be decomposed by two components, that is, local term weighting and global term weighting. Local term weighting schemes assign a weight to a term using local document information, such as term frequency and term co-occurrence in a target document. The local weight of a certain term varies depending on the document. Global term weighting schemes use global document information, such as document frequency and total term frequency, over all documents. The global weight of a certain term is fixed, that is, it does not depend on the target document in which the term occurs.

Representative term weighting schemes include TF-IDF, Okapi Best Matching 25 (BM25) [Robertson et al. 1994], and the recently proposed Term Weight-Inverse Document Frequency (TW-IDF) [Rousseau and Vazirgiannis 2013]. Each weight for term t in document $d \in D$ is computed as follows:

$$TF\text{-}IDF(t,d) = tf(t,d) \cdot \log \frac{|D|}{df(t)}$$

$$BM25(t,d) = \frac{(k_1+1)\cdot tf(t,d)}{k_1 \cdot \left(1-b+b\cdot \frac{|d|}{avdl}\right) + tf(t,d)} \cdot \log \frac{|D|}{df(t)}$$

$$TW\text{-}IDF(t,d) = \frac{tw(t,d)}{1-b+b\cdot \frac{|d|}{avdl}} \cdot \log \frac{|D|}{df(t)},$$

where $tf(t,d)$ is the term frequency of t in d, $df(t)$ is the document frequency of t over document set (corpus) D, $|D|$ is the cardinality of D (i.e., total number of documents in D), $|d|$ is the length of d (i.e., total number of words in d), $avdl$ is the average length of documents in D, $tw(t,d)$ is a graph-based tf-like function, and k_1 and b are constant values.[1] Whereas TF-IDF, Okapi BM25, and TW-IDF use very different local term weighting schemes, they use the same global term weighting scheme, that is, IDF [Jones 1972],

$$IDF(t) = \log \frac{|D|}{df(t)}. \tag{1}$$

IDF is adopted in such schemes due to its simplicity and robustness. The simplicity is easily understandable from Equation (1). In addition to simplicity, IDF has proven to be effective and robust [Aizawa 2003; Greiff 1998; Metzler 2008; Papineni 2001; Robertson 2004]. Other global term weighting schemes include Residual IDF (RIDF) [Church and Gale 1995] and gain [Papineni 2001]. Nonetheless, IDF has been used in many cases due to its simplicity (i.e., it is easy to use) and robustness (i.e., it works reasonably well for most applications).

One of the main drawbacks of IDF is that it cannot handle N-grams[2] for $N > 1$ or phrases that are composed of two or more words. IDF gives more weight to terms that occur in fewer documents. However, phrases occur in fewer documents when their collocations are less common. Consequently, uncommon phrases are unintentionally assigned disproportional weight. For example, the estimated document frequencies of "Osaka University" and "Osaka be" using Google Search[3] were 1,890,000 and 6,160,

[1] Reasonable values are to set k_1 to a value between 1.2 and 2, and $b = 0.75$ [Manning et al. 2008].
[2] We use *N-gram* to indicate word N-grams in this article.
[3] Results were retrieved on November 9, 2014.

respectively. Consequently "Osaka be" was assigned significantly greater IDF weight. Thus, the definition of IDF conflicts with the definition of good phrases.

Aside from term weighting schemes, a considerable number of studies have investigated extracting phrases, also referred to as Multiword Expressions (MWE). Pointwise Mutual Information (PMI) [Church and Hanks 1990] and Multiword Expression Distance (MED) [Bu et al. 2010] are representative schemes to measure how likely a sequence of consecutive words is to compose a phrase. The score simply indicates the compositionality of words, that is, it does not indicate how important or characteristic the composed phrase is. To the best of our knowledge, there is no theoretical explanation that deals with both term weighting and multiword expression extraction. To date, weighting terms of any length requires heuristics.

This article tackles bridging the theoretical gap between term weighting and multiword expression extraction. In particular, we connect IDF and MED in the information distance [Bennett et al. 1998] space. Information distance is a universal metric defined by Kolmogorov complexity [Kolmogorov 1963; Li and Vitányi 2009]. MED is a theoretically justified information distance-based metric for multiword expression extraction that works better than PMI and several heuristic measures. We prove that the IDF of a term is identical to the distance between the term and the empty string in the information distance space where Kolmogorov complexity is approximated using web documents and Shannon-Fano coding. Based on the findings, we design *N-gram IDF*, a global term weighting scheme that can handle N-grams of any length. N-gram IDF only requires the number of documents containing all constituent words composing an N-gram. N-gram IDF can weight phrases (i.e., N-grams for $N > 1$) and words in a single theoretical scheme, thereby enabling us to compare weights among words and phrases. Using N-gram IDF, we can detect *dominant N-grams* among overlapping N-grams and extract key terms of any length from texts without using NLP techniques.

Computing the N-gram IDF weight, that is, counting the number of documents containing multiple words (the so-called intersection query) for each N-gram is a nontrivial issue, because it is computationally expensive compared to single-word queries. This article examines how to solve this computational problem. First, we implement an exact computation method by combining two string processing algorithms, that is, enhanced suffix arrays and wavelet trees. Because the exact computation of intersection queries is computationally expensive [Gagie et al. 2012], we propose an approximation method that is several orders of magnitude faster than the exact method. N-gram IDF weight errors computed using the proposed approximation method can be estimated analytically.

We verify the robustness of N-gram IDF with applications that must handle N-grams of any length. Supervised and unsupervised key term extraction and web search query segmentation are employed, because their core task is to select meaningful N-grams among all N-grams in texts. We also show that the proposed approximation method for computing the N-gram IDF weights demonstrates stable performance. Furthermore, we evaluate N-gram IDF with languages other than English to support the universality of our N-gram weighting scheme. We conduct experiments on unsupervised key term extraction in Spanish, German, French, and Japanese.

The remainder of this article is organized as follows. Section 2 describes related work on term weighting and multiword expression extraction. Section 3 presents a justification of IDF based on Kolmogorov complexity and information distance. Based on this justification, Section 4 illustrates N-gram IDF, a modified IDF scheme to handle N-grams of any length. Section 5 proposes an efficient method to compute N-gram IDF weights for all N-grams using string processing algorithms. Section 6 shows various empirical evaluations of N-gram IDF and the computation method. Section 7 concludes this article.

2. RELATED WORK

Term weighting schemes are vital for text-related applications and thus have been well studied. TF-IDF [Salton and McGill 1983] is the most popular (general-purpose) term weighting scheme among representative schemes, followed by Okapi BM25 [Robertson et al. 1994]. In addition to the effectiveness of TF-IDF in various applications, such as information retrieval [Wu et al. 2008], document clustering [Fung et al. 2003], key term extraction [Hasan and Ng 2010], and object matching in videos [Sivic and Zisserman 2003], many theoretical explanations [Aizawa 2003; Hiemstra 2000; Robertson 2004; Roelleke and Wang 2008] have encouraged researchers and developers to employ TF-IDF. Based on or inspired by TF-IDF, heuristically improved schemes, such as TW-IDF [Rousseau and Vazirgiannis 2013], have been proposed.

The most popular global term weighting scheme is IDF [Jones 1972], which has been adopted by TF-IDF, BM25, and TW-IDF. There are many alternatives, such as x^I [Bookstein and Swanson 1974], z-measure [Harter 1975], RIDF [Church and Gale 1995], and gain [Papineni 2001]. However, IDF is still a de facto standard global term weighting scheme due to its simplicity and robustness compared to the alternatives. In addition to the simplicity of only computing $\log \frac{|D|}{df(t)}$, IDF has many justifications that back up its robustness [Aizawa 2003; Greiff 1998; Metzler 2008; Papineni 2001; Robertson 2004]. In other words, it is difficult to outperform IDF without using heuristics. Some studies have reported that RIDF was superior to IDF for certain applications [Orăsan et al. 2004; Rennie and Jaakkola 2005]. This indicates that RIDF can be a good alternative to IDF, although performance with other applications or datasets is not guaranteed due to its heuristic nature. Specialized global term weighting schemes, such as Inverse Corpus Frequency [Reed et al. 2006], which was designed to analyze text streams, and Relevant Frequency [Lan et al. 2009], which is used for text classification, can also be used in target applications.

The number of alternatives to IDF indicates that it has some limitations. Among them, a critical unsolved issue is its inability to handle phrases. Handling phrases requires measuring word compositionality, that is, the extent to which collocation is common, as well as term weighting. Robertson [2004] suggested that the sum of the IDF weights of words can be interpreted theoretically to weight phrases (or a set of words). However, this is still simply a term weighting scheme that does not include measurement of word compositionality.

The measurement of word compositionality, or MWE extraction, has been studied for a long time. PMI [Church and Hanks 1990] and variations of PMI are popular and effective methods for measuring word compositionality [Bouma 2009; da Silva and Lopes 1999; Dias 2000; Schone and Jurafsky 2001; Zhang et al. 2009]. Theoretically defined PMI was originally developed to measure the association between two words. Among 84 measures, PMI demonstrated the best performance for measuring bigram compositionality [Pecina 2005]. PMI must be extended to cope with N-grams where $N > 2$. Some studies have extended PMI heuristically by computing the arithmetic average of every possible separation [Dias 2000; da Silva and Lopes 1999] or the best score among every possible separation [Schone and Jurafsky 2001]. Enhanced Mutual Information [Zhang et al. 2009] is another extension of PMI that measures the cohesion of an N-gram using the frequency of each word. Symmetric Conditional Probability (SCP) [da Silva and Lopes 1999] measures bigram compositionality by multiplying the conditional probability of each word given the other. SCP handles N-grams where $N > 2$ using the arithmetic average.

MED [Bu et al. 2010] is a theoretically justified measure of the word (non-) compositionality for N-grams where $N > 1$ based on information distance [Bennett et al. 1998]. It has proven to be better than the heuristic extensions of PMI and SCP described above. Since MED is closely related to our work, we describe it in detail in Section 3. At this time, we can conclude that MED is the most promising measure of word compositionality based on a solid theory.

As indicated by the literature, term weighting and multiword expression extraction to measure the compositionality of consecutive words have been studied separately. However, no theoretical explanation that can handle both problems has been presented. In this work, we first give a theoretical explanation that connects IDF, which is a global term weighting scheme, and MED, which is a measure for multiword expression extraction.

3. ANOTHER JUSTIFICATION OF IDF

In this section, we explain Kolmogorov complexity, information distance, and MED to reveal the relationship between IDF and information distance.

3.1. Kolmogorov Complexity

Kolmogorov complexity [Kolmogorov 1963; Li and Vitányi 2009] is a measure of the randomness of a (bit) string. It is also known as descriptive complexity, algorithmic entropy, or program-size complexity, that is, the minimum resources required to describe a string on a universal Turing machine. We define $K(x)$ as the Kolmogorov complexity of string x. $K(x)$ is the length of the shortest program that outputs x. For example, given strings x_1 and x_2,

$$\begin{aligned} x_1 &= \text{“010101010101010100”} \\ x_2 &= \text{“011101100010110010”}, \end{aligned}$$

x_1 can be shortened to “01” ×8+ “00,” whereas x_2 seems difficult to shorten. Thus, we can estimate that $K(x_1)$ is smaller than $K(x_2)$. Because the exact value of the Kolmogorov complexity is not computable, it is usually approximated using compression algorithms [Cilibrasi and Vitányi 2005] or web documents [Cilibrasi and Vitányi 2007].

We also define $K(x|y)$ as the conditional Kolmogorov complexity of string x given another string y. $K(x|y)$ is the length of the shortest program that outputs x from input y. Given that ϵ is the empty string, $K(x)$ can be written as $K(x|\epsilon)$. Given strings x_1, x_2, and y such that

$$\begin{aligned} x_1 &= \text{“010101010101010100”} \\ x_2 &= \text{“011101100010110010”} \\ y &= \text{“01110110001011001”}, \end{aligned}$$

x_2 can be described efficiently as $y+$ “0” using y. Consequently, $K(x_2|y)$ might be slightly smaller than $K(x_1|y)$. Note that $K(x, y)$, that is, the Kolmogorov complexity of the concatenated string of strings x and y, is expressed as

$$K(x, y) = K(x|y) + K(y) = K(y|x) + K(x) \tag{2}$$

up to an additive logarithmic term. Equation (2) intuitively means that one string can be reused to describe the other.

3.2. Information Distance

Information distance [Bennett et al. 1998] is a universal metric defined by Kolmogorov complexity. It is an application-independent, unique objective distance that is equivalent to distance in the physical world. Essentially, it is the energy cost required to transform one string to another. According to Landauer's principle [Landauer 1961], irreversibly processing one bit of information costs $1\mathcal{KT} \cdot ln(2)$, where $\mathcal{K}$ is the Boltzmann constant and $\mathcal{T}$ is the absolute temperature in kelvin. Based on Landauer's principle and Kolmogorov complexity, information distance $E(x, y)$ between two strings x and y is defined as

$$E(x, y) = \max(K(x|y), K(y|x)) \tag{3}$$

up to an additive logarithmic term. Equation (3) is transformed into the following equation using Equation (2):

$$E(x, y) = K(x, y) - \min(K(y), K(x)). \tag{4}$$

Information distance has proven to be a metric, or distance function, that is, it satisfies non-negativity, identity of indiscernibles, symmetry, and triangle inequality, which are respectively represented by the following formulas:

$$\begin{aligned} &E(x, y) \geq 0 \\ &E(x, y) = 0 \Leftrightarrow x = y \\ &E(x, y) = E(y, x) \\ &E(x, z) \leq E(x, y) + E(y, z). \end{aligned}$$

Moreover, information distance has been shown to be universal or optimal. Distance $\mathcal{D}(x, y)$ is admissible (i.e., an upper semi-computable and normalized metric) if it satisfies the following density condition (Kraft's inequality):

$$\sum_{y:y \neq x} 2^{-\mathcal{D}(x,y)} \leq 1, \quad \sum_{x:x \neq y} 2^{-\mathcal{D}(x,y)} \leq 1.$$

The density condition restricts the number of objects within a given distance from an object. When $\mathcal{D}(x, y)$ is admissible, there is a constant C for all x and y, and

$$E(x, y) \leq \mathcal{D}(x, y) + C.$$

Thus, $E(x, y)$ minorizes every admissible distance $\mathcal{D}(x, y)$ up to an additive constant, indicating that information distance is universal.

Information distance is generally used to measure similarity between two objects. Because the exact value of the information distance is not computable, similarly to Kolmogorov complexity, it is computed using approximations, such as Normalized Compression Distance (NCD) [Cilibrasi and Vitányi 2005] and Normalized Google Distance (NGD) [Cilibrasi and Vitányi 2007]. NCD measures the compression size of the concatenated string of x and y versus that of each string. NGD is intended for texts. NGD counts the number of web pages containing both terms x and y versus the number of web pages containing each term. Cilibrasi and Vitányi [2007] reported that the distance estimated by NGD was stable for the growing web. It can be presumed that the stability comes from the universality of information distance.

3.3. Multiword Expression Distance

MED [Bu et al. 2010] is a universal metric for measuring word compositionality based on information distance. MED computes the information distance between the context and semantic of an N-gram. Inspired by Cilibrasi and Vitányi [2007], Bu et al. [2010] defined the context of an N-gram as a set of web pages containing the given N-gram and the semantic of an N-gram as a set of web pages containing each constituent word comprising the N-gram. Obviously, the semantic of an N-gram subsumes the context of the N-gram. For example, the semantic of "football player" includes both web pages containing "football player" and those containing "football" and "player."

Formulating MED according to Bu et al. [2010], we denote w as a word (unigram), W as a set of all words, g as an N-gram, $G \equiv W^+$ as a set of all N-grams, D as a set of all web pages (documents), t as a search term that is an N-gram or the conjunction of (a set of) search terms, and T as a set of all search terms (i.e., $G \subset T$). Here let $\phi : T \to 2^D$ be the context function that maps a search term t to a set of web pages containing all N-grams in t, denoted by $\phi(t)$. Let $\theta : G \to T$ be the function that maps an N-gram $g = w_1 \cdots w_N$ to $\bigwedge_{i=1}^{N} w_i$, the conjunction of words composing g (denoted $\theta(g)$). Let $\mu : G \to 2^D$ be the semantic function that is a composite function $\phi \circ \theta$, that is, $\mu(g) = \phi(\theta(g))$. From the definitions, we obtain $\phi(g) \subseteq \mu(g)$. Given an N-gram g, MED is defined as the information distance between $\phi(g)$, the context of g, and $\mu(g)$, that is, the semantic of g. Using Equation (4), MED is formulated as follows:

$$MED(g) = E(\phi(g), \mu(g)) = K(\phi(g), \mu(g)) - \min(K(\phi(g)), K(\mu(g))). \tag{5}$$

To compute the Kolmogorov complexity $K(x)$ approximately, MED utilizes Shannon-Fano coding to encode the probability of x. We assume that all web pages are equiprobable, that is, the probability that a web page is selected is $\frac{1}{|D|}$. Let $p : \phi(T) \to [0, 1]$ be the context probability function where $\phi(T)$ is a set of all contexts, that is, $\phi(T) \equiv \{x | \exists y \in T, x = \phi(y)\}$. Given that c is a context (a set of web pages) obtained from search term t, the probability that a randomly chosen web page contains t is $\frac{|c|}{|D|}$ [Li and Vitányi 2009]. Similarly, the probability of context c obtained from search term t is proportional to $|c|$. It is specifically defined as

$$p(c) = \frac{|c|}{M}, \tag{6}$$

where the denominator $M = \sum_{c_i \in \phi(T)} |c_i|$ rather than $|D|$ such that $p(c)$ can be a valid probability function. Consequently, the Kolmogorov complexity K can be approximated using the Shannon-Fano code length associated with p [Li and Vitányi 2009],

$$K(x) \approx -\log p(x), \tag{7}$$

$$K(x, y) \approx -\log p(x, y). \tag{8}$$

Using Equations (6), (7), and (8), Equation (5) is approximated as

$$\begin{aligned} MED(g) &\approx \max(\log p(\phi(g)), \log p(\mu(g))) - \log p(\phi(g), \mu(g)) \\ &= \max(\log |\phi(g)|, \log |\mu(g)|) - \log M - \log |\phi(g) \cap \mu(g)| + \log M \\ &= \max(\log |\phi(g)|, \log |\mu(g)|) - \log |\phi(g) \cap \mu(g)|. \end{aligned} \tag{9}$$

Since $\phi(g) \subseteq \mu(g)$, Equation (9) is rewritten as

$$MED(g) \approx \log |\mu(g)| - \log |\phi(g)| = \log \frac{|\mu(g)|}{|\phi(g)|}. \tag{10}$$

Equation (10) can be computed using the cardinality of $\phi(g)$ and $\mu(g)$. Specifically, $|\phi(g)|$ is the document frequency of g and $|\mu(g)|$ is the document frequency of $\theta(g)$, that is, a conjunction of words composing g. In the literature [Bu et al. 2010], $|\phi(g)|$ and $|\mu(g)|$ are estimated by using a general search engine. The document frequency of $\theta(g)$ is obtained from a "logical AND" query for each word w_i in g.

3.4. Inverse Document Frequency

Here, we reveal the relationship between IDF and information distance based on the above discussions. Just like Equation (1), we define IDF of N-gram g as follows:

$$IDF(g) = \log \frac{|D|}{df(g)} = \log \frac{|D|}{|\phi(g)|}. \tag{11}$$

Note that the mathematical forms of Equation (11) for IDF and Equation (10) for MED are analogous. The difference is that, in IDF, the numerator in the logarithmic term is

$|D|$ and in MED it is $|\mu(g)|$. Are there any relationships between IDF and information distance? Here, we investigate this question.

We denote the empty string, or the zero-gram, by ϵ. We define $\epsilon \in G$ and $G \equiv W^*$. $\phi(\epsilon)$ is a set of web pages containing ϵ. Clearly, ϵ is contained in all web pages. Therefore, $\phi(\epsilon)$ is equal to D, that is, the set of all web pages. We then derive the information distance between N-gram g and the empty string ϵ. $E(\phi(g), \phi(\epsilon))$ is defined as

$$E(\phi(g), \phi(\epsilon)) = K(\phi(g), \phi(\epsilon)) - \min(K(\phi(g)), K(\phi(\epsilon))). \tag{12}$$

As well as the derivation of MED, Equation (12) can be transformed into

$$\begin{aligned} E(\phi(g), \phi(\epsilon)) &\approx \max(\log p(\phi(g)), \log p(\phi(\epsilon))) - \log p(\phi(g), \phi(\epsilon)) \\ &= \max(\log |\phi(g)|, \log |\phi(\epsilon)|) - \log M - \log |\phi(g) \cap \phi(\epsilon)| + \log M \\ &= \max(\log |\phi(g)|, \log |\phi(\epsilon)|) - \log |\phi(g) \cap \phi(\epsilon)|. \end{aligned} \tag{13}$$

Because apparently $\phi(g) \subseteq \phi(\epsilon)$, Equation (13) becomes

$$E(\phi(g), \phi(\epsilon)) \approx \log |\phi(\epsilon)| - \log |\phi(g)| = \log \frac{|\phi(\epsilon)|}{|\phi(g)|} = \log \frac{|D|}{|\phi(g)|} = \mathit{IDF}(g).$$

Finally, we obtain the relationship between IDF and information distance, that is, IDF of an N-gram g is equal to the information distance between $\phi(g)$ and $\phi(\epsilon)$.

In the following, we summarize our findings about IDF and information distance. In the information distance space in which the Kolmogorov complexity is approximated using web documents and Shannon-Fano coding, the IDF of a term is equal to the distance between the term and the empty string, which is normally the base of the Kolmogorov complexity. Therefore, the information distance can be used as the global weight of terms. The farther a term is from the empty string in the information distance space, the larger its weight. Thus, we can design term weighting schemes using the information distance space. Our findings are substantially useful, because it becomes far easier to combine term weighting with MED, an information distance-based metric for multiword expression extraction. In the next section, we show a theoretically justified combination of IDF and MED for weighting N-grams of any N.

4. N-GRAM IDF

We propose a global term weighting scheme that can handle terms of any length, that is, N-grams where $N \geq 1$, based on the information distance. In Section 3, we revealed that the IDF of an N-gram is equal to the information distance between the N-gram and the empty string when the Kolmogorov complexity is approximated using web documents and Shannon-Fano coding. Similarly, the MED of an N-gram is the information distance between the N-gram and a set of words composing the N-gram. Figure 1 shows the relationship between IDF and MED in the information distance space. In this space, N-gram g, a set of words $\theta(g)$, and the empty string ϵ are represented as the sets of web pages $\phi(g)$, $\mu(g)$, and $\phi(\epsilon)$, respectively. Note that the equality holds in the triangle inequality as follows:

$$E(\phi(g), \mu(g)) + E(\mu(g), \phi(\epsilon)) = \log \frac{|\mu(g)|}{|\phi(g)|} + \log \frac{|D|}{|\mu(g)|} = \log \frac{|D|}{|\phi(g)|} = E(\phi(g), \phi(\epsilon)).$$

Here, the challenge is to theoretically connect the term weighting and multiword expression extraction, both of which have been studied separately for a long time. IDF gives too much weight to an N-gram for $N > 1$ when the collocation of the N-gram is less common. In this case, MED should also be large, since it exactly measures how uncommon the collocation of the N-gram is. Taken together, it seems reasonable to give more weight to an N-gram when the IDF is large and the MED is small. We can

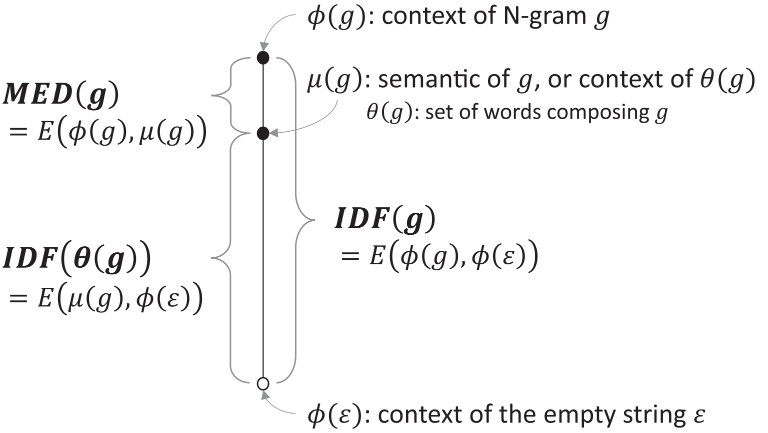

$\phi(g)$: context of N-gram g
$\mu(g)$: semantic of g, or context of $\theta(g)$
$\theta(g)$: set of words composing g
$MED(g) = E(\phi(g), \mu(g))$
$IDF(g) = E(\phi(g), \phi(\varepsilon))$
$IDF(\theta(g)) = E(\mu(g), \phi(\varepsilon))$
$\phi(\varepsilon)$: context of the empty string ε

Fig. 1. Relationship between IDF and MED in the information distance space where the Kolmogorov complexity is approximated using web documents and Shannon-Fano coding. $\phi(g)$, $\mu(g)$, and $\phi(\epsilon)$ are on a line, that is, the equality holds in triangle inequality.

consider two such schemes from Figure 1: $IDF(\theta(g))$ and $IDF(\theta(g)) - MED(g)$. Both are related to N-gram IDF in the sense that the IDF of $\theta(g)$, a set of words composing N-gram g, is used.

4.1. Direct N-gram IDF

We define $IDF(\theta(g))$ as a direct N-gram IDF $NIDF_d(g)$. It directly defines the weight of an N-gram by its semantic, that is, a set of constituent words. As can be seen in Figure 1, it is equivalent to $IDF(g) - MED(g)$. In other words, it is the information distance between $\theta(g)$ and the empty string ϵ. $NIDF_d(g)$ is computed as follows:

$$NIDF_d(g) = IDF(\theta(g)) = \log \frac{|D|}{|\mu(g)|} = \log \frac{|D|}{df(\theta(g))}. \tag{14}$$

When $N = 1$, $NIDF_d(g)$ corresponds to $IDF(g)$, because $\theta(g) = g$, that is, $df(\theta(g)) = df(g)$. The direct N-gram IDF weight grows monotonically as N increases. Therefore, overlapping N-grams of different N cannot be compared solely using the direct N-gram IDF weights. Direct N-gram IDF may be useful when combined with other weights or measures, such as the total term frequency in a corpus.

4.2. Indirect N-gram IDF

We define $IDF(\theta(g)) - MED(g)$ as an indirect N-gram IDF $NIDF_i(g)$. We can explain this by "how much the distance from the empty string to the semantic of g can be shortened by providing the context of g." In Section 3, we revealed that the distance in the information distance space represents the weight of a term. Here, we measure the distance in an indirect manner. The distance shortened by observing N-gram g indicates how much weight g has. $NIDF_i(g)$ is computed as follows:

$$\begin{aligned} NIDF_i(g) &= IDF(\theta(g)) - MED(g) \\ &= \log \frac{|D|}{|\mu(g)|} - \log \frac{|\mu(g)|}{|\phi(g)|} = \log \frac{|D| \cdot |\phi(g)|}{|\mu(g)|^2} = \log \frac{|D| \cdot df(g)}{df(\theta(g))^2}. \end{aligned} \tag{15}$$

Note that $NIDF_i(g)$ corresponds to $IDF(g)$ when $N = 1$.

Table I. Examples of Indirect N-gram IDF. Dominant N-grams Are Indicated in Bold with an Asterisk (Details of the Dominant N-grams for the Text in the First Column Are Represented in Figure 2)

Alice's Adventures in Wonderland—Kindle edition by Lewis Carroll		Fossil fuels must be phased out "almost entirely" by 2100 to avoid dangerous climate change	
***kindle edition**	12.043	***fossil fuels**	11.211
kindle	11.653	2100	10.772
***alice s adventures in wonderland**	11.496	fuels	9.752
adventures in wonderland	10.906	***phased**	9.391
s adventures in wonderland	10.804	phased out	9.291
wonderland	9.670	fossil	8.332
***lewis carroll**	9.498	***dangerous climate change**	8.249
alice s adventures	9.385	climate change	7.432
alice s adventures in	9.348	be phased out	6.828
in wonderland	8.762	by 2100	6.814
carroll	8.152	be phased	6.783
by lewis carroll	7.461	dangerous	6.749
alice	7.234	climate	6.575
adventures	7.101	dangerous climate	6.549
kindle edition by	6.739	***almost entirely**	6.118
lewis	6.192	***avoid**	6.063
edition	4.836	entirely	5.973
adventures in	4.280	to avoid	5.831
s adventures	3.586	2100 to	5.031
alice s	3.507	***must**	4.703
s adventures in	2.255	change	4.646
by lewis	1.768	almost	4.469
s	1.030	fuels must	4.283
by	0.820	must be phased	4.013
in	0.154	must be phased out	4.010
edition by	−0.875	must be	3.831
		fuels must be	3.478
		avoid dangerous	2.998
		out	2.612
		to avoid dangerous	2.575
		entirely by	2.055
		almost entirely by	1.984
		be	1.860
		by	0.820
		to	0.505
		out almost	−2.826

4.3. Dominant N-gram

We found that indirect N-gram IDF (Equation (15)) is surprisingly stable for any N. Table I provides examples of N-gram IDF[4] for a couple of texts (the weight is computed in Section 6.1). An important feature of N-gram IDF is the comparability of weights among N-grams of different N. Therefore, we can establish the dominance relationship among the weights of all N-grams in a text. We define *dominant N-grams* as those with the maximum weight of at least one position (word) of a given text. Dominant N-grams are determined by the following simple algorithm.

[4] Henceforth, N-gram IDF refers to indirect N-gram IDF.

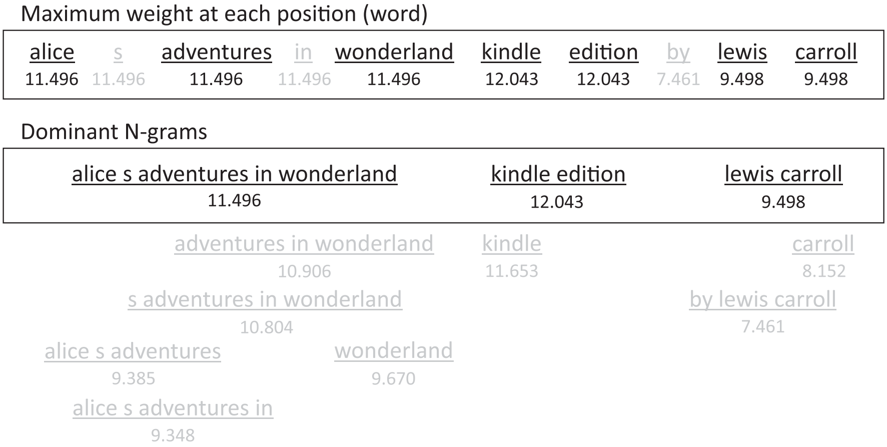


Fig. 2. Example of dominant N-grams for the text in the first column of Table I.

(1) Set the maximum weight at each position (word) by checking all weights of N-grams that overlap the position.
(2) For each position, select a dominant N-gram whose weight is equal to the maximum weight of the position.

Figure 2 illustrates dominant N-grams for the text in the first column of Table I. For example, the dominant N-gram at position “alice” is “alice s adventures in wonderland,” which has a maximum weight of 11.496. Stop words are ignored to determine dominant N-grams, for example, “by lewis carroll” has the maximum weight of position “by”; however, it is not a dominant N-gram. Overlapping N-grams rarely have the same weight. We must introduce some policies to manage such cases, for example, longest prefix match. The number of dominant N-grams does not exceed the length of the text, because one position of the text inevitably corresponds to a dominant N-gram. From Table I, we can confirm that N-gram IDF weights are stable regardless of N and dominant N-grams are reasonably determined across different N.

4.4. Comparison with Robertson’s Interpretation

Prior to the development of N-gram IDF, Robertson [2004] provided a theoretical interpretation of IDF for phrases (or a set of words). The sum of the IDF weights of words composing an N-gram can be interpreted based on a probabilistic model. Given that $P(t)$ is the probability that term t appears in a randomly chosen document from D, IDF is defined as

$$IDF(t) = \log \frac{|D|}{df(t)} = -\log P(t). \tag{16}$$

Assuming the occurrences of words in documents are statistically independent, we can estimate the IDF of a set of words composing N-gram g using Equation (16). Specifically, the IDF of a set of words $\theta(g) = \bigwedge_{i=1}^{N} w_i$ is computed as follows:

$$\begin{aligned} IDF(\theta(g)) &= IDF\left(\bigwedge_{i=1}^{N} w_i\right) = -\log P\left(\bigwedge_{i=1}^{N} w_i\right) \\ &= -\log\left(\bigwedge_{i=1}^{N} P(w_i)\right) = -\left(\sum_{i=1}^{N} \log P(w_i)\right) = \sum_{i=1}^{N} IDF(w_i). \end{aligned} \tag{17}$$

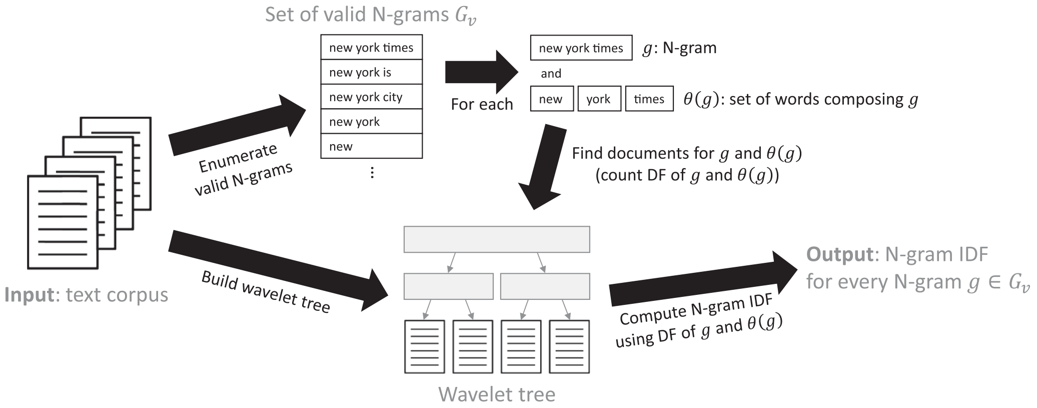

Set of valid N-grams G_v
new york times
new york is
new york city
new york
new
⋮
For each
new york times
g: N-gram
and
new
york
times
$\theta(g)$: set of words composing g
Enumerate valid N-grams
Find documents for g and $\theta(g)$ (count DF of g and $\theta(g)$)
Input: text corpus
Build wavelet tree
Wavelet tree
Compute N-gram IDF using DF of g and $\theta(g)$
Output: N-gram IDF for every N-gram $g \in G_v$

Fig. 3. Outline of N-gram IDF computation method.

When we remove the assumption that word occurrences are statistically independent, $IDF(\theta(g))$ cannot be derived using Equation (17) but instead becomes identical to direct N-gram IDF, as shown in Equation (14). Considering that the occurrences of words are not independent in real-world documents, N-gram IDF turns out to measure the IDF of a set of words according to a more realistic assumption. Interestingly, a similar discussion can be found in the literature. Bu et al. [2010] noted the relationship between MED and PMI, that is, the difference between MED and PMI is that MED considers dependency among word occurrences.

5. COMPUTATION METHODS

Computing N-gram IDF or MED involves *intersection* [Demaine et al. 2000] queries, that is, counting the number of documents that contain multiple words. Bu et al. [2010] adopted an ad hoc approach that utilizes a general web search engine to estimate the document frequency of a set of words. In addition, there is a straightforward way to count the document frequency of a set of words by building inverted indices for words and obtaining the intersection. When the target is a small number of N-grams, these approaches are reasonable. However, computing the document frequency of a set of constituent words for all N-grams is difficult, because handling intersection queries provably requires significant time compared to single or union queries [Demaine et al. 2000]. Therefore, we attempt to find efficient methods to perform intersection queries.

5.1. Exact Computation Method

5.1.1. Outline. First, we attempt to compute exact N-gram IDF weights for all N-grams. To solve the problem, we introduce two data structures that have been developed in the field of theoretical computer science, that is, enhanced suffix arrays [Abouelhoda et al. 2004] (Appendix A) and wavelet trees [Grossi et al. 2003] (Appendix B). Figure 3 outlines the method for computing N-gram IDF for all valid N-grams. Given a text corpus (a set of documents), the method enumerates valid N-grams (let G_v be a set of valid N-grams) by building an enhanced suffix array for a single concatenated text of the corpus. The method also builds a wavelet tree for documents. For each valid N-gram, the method computes the document frequencies of the N-gram and a set of words using the wavelet tree. Finally, N-gram IDF for every valid N-gram is computed using the document frequencies.

5.1.2. Intersection Query on Wavelet Tree. The core of the N-gram IDF computation method lies in the use of a wavelet tree to count the document frequency (the number

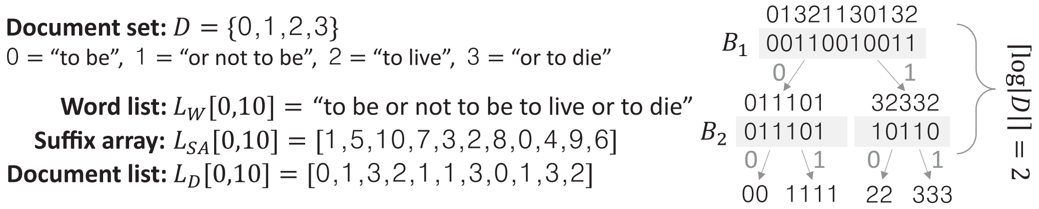

Document set: $D = \{0,1,2,3\}$
0 = "to be", 1 = "or not to be", 2 = "to live", 3 = "or to die"
Word list: $L_W[0,10]$ = "to be or not to be to live or to die"
Suffix array: $L_{SA}[0,10] = [1,5,10,7,3,2,8,0,4,9,6]$
Document list: $L_D[0,10] = [0,1,3,2,1,1,3,0,1,3,2]$
01321130132
B_1 00110010011
0
1
011101
32332
B_2 011101
10110
0
1
0
1
00 1111 22 333
$\lceil \log|D| \rceil = 2$

Fig. 4. Example of the wavelet tree for a document list.

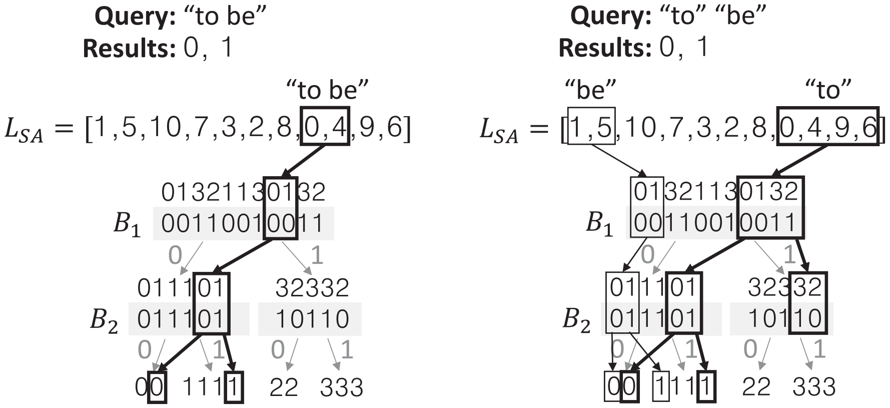


Fig. 5. Examples of the document listing on the wavelet tree constructed in Figure 4: (left) bigram query "to be" (right) query of a set of two words "to" and "be."

of documents) for a set of words, that is, handling intersection queries on the wavelet tree [Gagie et al. 2012]. The wavelet tree for documents is constructed as follows. Given a set of documents D as a single concatenated text $L_W[0, n-1]$ (i.e., a list of n words), we generate suffix array $L_{SA}[0, n-1]$, which is a list of indices (positions) for L_W sorted by their suffixes, that is, N-gram $g_v \in G_v$ appears in a single segment in L_{SA}. Then, we create list $L_D[0, n-1]$ of documents such that word $L_W[L_{SA}[i]]$ occurs in document $L_D[i]$, that is, index i in L_{SA} corresponds to index i in L_D. Thus, we can represent all documents containing N-gram $g_v \in G_v$ by a single segment in L_D. The wavelet tree is constructed for L_D. Each leaf node in the wavelet tree represents a distinct document in D.

Figure 4 shows an example of the wavelet tree for a document list. As can be seen, suffixes starting at the positions in L_{SA} ("be or not $\cdots$" at position 1, "be to live $\cdots$" at position 5, "die" at position 10, etc.) are sorted. Since all documents containing N-gram $g_v \in G_v$ are represented as a segment in L_{SA} and thus L_D, enumerating all documents is performed by accessing all symbols on the wavelet tree given the start and end indices of the segment (see Appendix B for how to access symbols on the wavelet tree).

Figure 5 illustrates the document listing using the wavelet tree described in Figure 4. The left example finds documents containing the bigram "to be," which occurs at $L_W[i]$ for $i \in L_{SA}[7, 8]$ (i.e., the start and end indices are 7 and 9). Figure 10 in Appendix A visually explains why "to be" is represented as the single segment $L_{SA}[7, 8]$ in the list of positions L_{SA}. Since index i in L_{SA} corresponds to index i in L_D, "to be" also corresponds to $L_D[7, 8]$. Because B_1 stores the first bit of L_D (see Appendix B), documents containing "to be" can be accessed starting from $B_1[7, 8]$ in the wavelet tree. The start and end indices at node 0 (1) in B_2 are found by counting the number of occurrences of bit 0 (1) in $B_1[0, 6]$ and $B_1[0, 8]$, respectively. If the start and end indices in a child node become

ALGORITHM 1: CountDF

```
   Input: Start indices $i_s[0, m-1]$, end indices $i_e[0, m-1]$, numbers of occurrences
          $o[0, m-1]$, depth $k = 1$, start position $p_s = 0$, end position $p_e = n$
   /* Initial input is a set of words (or N-grams) $w_0, \ldots, w_{m-1}$. Each word $w_j$ is
      converted into start index $i_s[j]$ and end index $i_e[j]$ in $L_{SA}$. $o[j]$ is set to 1
      except when $w_j$ redundantly appears in an N-gram.                                  */
   Output: Document frequency $df$
 1 if $k > \lceil \log |D| \rceil$ then // reach the leaf node
 2 |   return 1;
 3 end
 4 $df = 0,\ f_0 = \text{true},\ f_1 = \text{true}$;
 5 $p_b = p_s + rank_0(B_k, p_e) - rank_0(B_k, p_s)$;
 6 init $i_{s0}[0, m-1], i_{e0}[0, m-1], i_{s1}[0, m-1], i_{e1}[0, m-1]$;
 7 for $j = 0$ to $m-1$ do
 8 |   $i_{s0}[j] = rank_0(B_k, p_s + i_s[j]) - rank_0(B_k, p_s)$;
 9 |   $i_{e0}[j] = rank_0(B_k, p_s + i_e[j]) - rank_0(B_k, p_s)$;
10 |   $i_{s1}[j] = rank_1(B_k, p_s + i_s[j]) - rank_1(B_k, p_s)$;
11 |   $i_{e1}[j] = rank_1(B_k, p_s + i_e[j]) - rank_1(B_k, p_s)$;
12 |   if $i_{e0}[j] - i_{s0}[j] < o[j]$ then
13 |   |   $f_0 = \text{false}$;
14 |   end
15 |   if $i_{e1}[j] - i_{s1}[j] < o[j]$ then
16 |   |   $f_1 = \text{false}$;
17 |   end
18 end
19 if $f_0$ then // visit child node 0
20 |   $df = CountDF(i_{s0}, i_{e0}, o, p_s, p_b, k+1)$;
21 end
22 if $f_1$ then // visit child node 1
23 |   $df = df + CountDF(i_{s1}, i_{e1}, o, p_b, p_e, k+1)$;
24 end
25 return $df$;
```

the same, then there is no corresponding bit in the node. Finally, we obtain documents 0 and 1 by enumerating leaf nodes that are reached. The right example in Figure 5 handles an intersection query, which is our main problem, in a manner similar to the left example. Child nodes are visited only if there are corresponding bits for both “to” and “be.” We simply count the number of leaf nodes that are reached.

Algorithm 1 (an extension of Algorithm 3 in Appendix B) details a recursive function used to count the document frequency on a wavelet tree. The initial input is a search term consisting of m ($m \geq 1$) unique words (or N-grams). w_j, that is, word with index j ($0 \leq j < m$), is converted into start index $i_s[j]$ and end index $i_e[j]$ in L_{SA} (thus, they are the start and end indices in L_D). $o[j]$ is usually 1; however, it becomes greater than 1 when N-grams contain duplicate words (e.g., the movie title “run fatboy run” contains two instances of the word “run”). Depth k, start position p_s, and end position p_e are initially set to 1, 0, and n, respectively, indicating the root node $B_1[0, n-1]$ of the wavelet tree. For each word, the next start and end indices in child nodes 0 and 1 at depth $k+1$ are computed using the *rank* function (Lines 8–11). If the length between the next start and end indices of word of index j is not less than $o[j]$ for every j (Lines 12–17), then the child node is visited. Algorithm 1 recursively calls the function to visit child nodes (Lines 19–24). It increments document frequency df when it reaches a leaf node (Lines 1–3). For the last time, the number of times a leaf node is visited is equal to the document frequency for the initial inputs.

To count the document frequency of a set of words composing N-gram g using Algorithm 1, we must convert each word to start and end indices in L_{SA}. Practically, the start and end indices can be easily found by traversing the tree structure represented by an enhanced suffix array (Appendix A). For example, the word "to" is converted into the segment between seventh and tenth (containing positions 0, 4, 9, and 6) in Figure 10. $o[j]$, that is, the number of occurrences, is set to 1, except when g contains duplicate words (in this case, $o[j]$ for that word is set to the number of duplications).

5.2. Approximate Computation Method

While the computational cost of the exact computation method (Section 5.1.2) approaches the lower bound [Gagie et al. 2012], it still takes significant time to process a large corpus, for example, all of English Wikipedia, that contains numerous valid N-grams, because $df(\theta(g))$ grows by $O(|D|)$ for many N-grams along with corpus size $|D|$. In other words, alternation complexity α [Gagie et al. 2012] is $O(|D|)$ and time complexity $O(N \cdot \alpha \cdot \log \frac{|D|}{\alpha})$ [Gagie et al. 2012] approaches $O(N \cdot |D|)$. Note that the number of all valid N-grams also becomes $O(|D|)$. The total time complexity turns out to be approximately $O(|D|^2)$ (given that N is relatively small for nearly all N-grams).

To solve the problem, we propose a subset-based approximation method that is several orders of magnitude faster than the exact method. The proposed approximation method can estimate the error of the (N-gram) IDF weights using statistical distribution. The performance of approximate weights at an application level is nearly the same as that of exact weights. We describe processing time, practical error distributions, and performance evaluations in Section 6.5.

5.2.1. Analysis of Approximation. The basic idea of the approximation is to use a subset of the corpus to estimate the document frequency. Given a (random) subset D_s of corpus D (i.e., $D_s \subset D$), the expected document frequency $\langle df_s(g)\rangle$ of N-gram g in D_s is

$$\langle df_s(g)\rangle = \frac{|D_s|}{|D|} \cdot df(g).$$

Conversely, when we observe document frequency $df_s(g)$ in D_s, the expected document frequency $\langle df(g)\rangle$ of N-gram g in D is

$$\langle df(g)\rangle = \frac{|D|}{|D_s|} \cdot df_s(g).$$

The IDF of g in D is then expected from $df_s(g)$ as follows:

$$\langle IDF(g)\rangle = \log \frac{|D|}{\langle df(g)\rangle} = \log \frac{|D|}{\frac{|D|}{|D_s|} \cdot df_s(g)} = \log \frac{|D_s|}{df_s(g)}. \tag{18}$$

The question here is as follows: How does the expected IDF $\langle IDF(g)\rangle$ differ from the actual IDF $IDF(g)$? We introduce Poisson distribution to answer this question. Poisson distribution is a discrete probability distribution that expresses the probability of a given number of times a specific event occurs in a fixed space when its average rate in the fixed space is known and the occurrence of each event is independent [Haight 1967]. Poisson distribution has a single parameter λ, that is, the average rate in the fixed space, which is the mean and variance of the distribution. The probability mass function of discrete random variable $X (x = 0, 1, 2, \ldots)$ having Poisson distribution with parameter λ is given by

$$P_\lambda(X = x) = \frac{\lambda^x e^{-\lambda}}{x!}.$$

Table II. Confidence Intervals for the Mean of a Poisson Distribution

	Percent confidence					
	90%		95%		99%	
k	Lower limit	Upper limit	Lower limit	Upper limit	Lower limit	Upper limit
1	0.05	4.74	0.03	5.57	0.01	7.43
5	1.97	10.51	1.62	11.67	1.08	14.15
10	5.43	16.96	4.80	18.39	3.72	21.40
20	13.25	29.06	12.22	30.89	10.35	34.67
50	38.96	63.29	37.11	65.92	33.66	71.27
100	84.14	118.08	81.36	121.63	76.12	128.76

It can be transformed as

$$P_\lambda(X = x) = \exp(x \log \lambda - \lambda - \log \Gamma(x + 1)),$$

where $\log \Gamma$ is the natural logarithm of the gamma function, which is numerically stable.[5]

The document frequency in a fixed size of subset corpus D_s follows Poisson distribution with parameter $\lambda_s = \langle df_s(g) \rangle$ given that $\langle df_s(g) \rangle \ll |D_s|$ holds for nearly all N-grams,[6] that is, observed document frequency $df_s(g)$ is a sample from Poisson distribution. The difference between $df_s(g)$ and $\langle df_s(g) \rangle$ can be estimated analytically using the confidence intervals. Table II shows some confidence intervals for the mean of a Poisson distribution [van Belle et al. 2004]. When we observe the event exactly k times, the true mean falls into the range between its lower and upper limits with the percent confidence. Consider the following example. We observe document frequency $df_s(g) = 20$ (i.e., $k = 20$). Expected document frequency $\langle df_s(g) \rangle$ is then between 10.35 and 34.67 with 99% confidence.

The difference between the observed and expected document frequencies in a subset of the corpus determines the IDF error. We denote $\lambda_s^{(L)}$ and $\lambda_s^{(U)}$ as the lower and upper limits of λ_s, respectively, given observed document frequency $df_s(g)$ in D_s. The lower and upper limits of IDF for N-gram g, that is, $IDF^{(L)}(g)$ and $IDF^{(U)}(g)$, are defined as follows:

$$IDF^{(L)}(g) = \log \frac{|D_s|}{\lambda_s^{(U)}}, \tag{19}$$

$$IDF^{(U)}(g) = \log \frac{|D_s|}{\lambda_s^{(L)}}. \tag{20}$$

From Equations (18), (19), and (20), the underestimated and overestimated errors of IDF are calculated as follows:

$$\langle IDF(g) \rangle - IDF^{(U)}(g) = \log \frac{|D_s|}{df_s(g)} - \log \frac{|D_s|}{\lambda_s^{(L)}} = \log \frac{\lambda_s^{(L)}}{df_s(g)},$$

$$\langle IDF(g) \rangle - IDF^{(L)}(g) = \log \frac{|D_s|}{df_s(g)} - \log \frac{|D_s|}{\lambda_s^{(U)}} = \log \frac{\lambda_s^{(U)}}{df_s(g)}.$$

Since $\lambda_s^{(L)}$ and $\lambda_s^{(U)}$ are determined by observed document frequency $df_s(g)$, the IDF error is solely subject to $df_s(g)$. For example, when we observe document frequency

[5] The natural logarithm of the gamma function is supported in major standard libraries and tools, for example, C/C++, python, MATLAB, and Microsoft Excel.

[6] We can ignore the rare cases when $\langle df_s(g) \rangle$ is close to $|D_s|$, because the observed document frequency approaches $|D_s|$.

$df_s(g) = 20$, the underestimated error of IDF is $\log \frac{10.35}{20} \approx -0.95$ and the overestimated error is $\log \frac{34.67}{20} \approx 0.79$ with 99% confidence. Note that the size of subset corpus $|D_s|$ does not affect the IDF error.

The same estimate can be applied to direct N-gram IDF $NIDF_d(g)$. For indirect N-gram IDF $NIDF_i(g)$, the error is doubled. Given that $df(g)$ is computed exactly, the expected N-gram IDF $\langle NIDF_i(g)\rangle$ is computed as

$$\langle NIDF_i(g)\rangle = \log \frac{|D| \cdot df(g)}{\langle df(\theta(g))\rangle^2} = \log \frac{|D| \cdot df(g)}{\left(\frac{|D|}{|D_s|} \cdot df_s(\theta(g))\right)^2} = \log \frac{|D_s|^2 \cdot df(g)}{|D| \cdot df_s(\theta(g))^2},$$

where $df_s(\theta(g))$ is the observed document frequency of a set of constituent words composing N-gram g in D_s. Given that $\langle df_s(\theta(g))\rangle$ is the expected document frequency in a subset corpus of size $|D_s|$, $df_s(\theta(g))$ is a sample from the Poisson distribution with parameter $\lambda_s = \langle df_s(\theta(g))\rangle$. Let $\lambda_s^{(L)}$ and $\lambda_s^{(U)}$ be the lower and upper limits of λ_s, respectively, given observed document frequency $df_s(\theta(g))$ in D_s. The lower and upper limits of N-gram IDF are defined as follows:

$$NIDF_i^{(L)}(g) = \log \frac{|D_s|^2 \cdot df(g)}{|D| \cdot \left(\lambda_s^{(U)}\right)^2},$$

$$NIDF_i^{(U)}(g) = \log \frac{|D_s|^2 \cdot df(g)}{|D| \cdot \left(\lambda_s^{(L)}\right)^2}.$$

Finally, the underestimated and overestimated errors of N-gram IDF become the following:

$$\langle NIDF_i(g)\rangle - NIDF_i^{(U)}(g) = \log \frac{\left(\lambda_s^{(L)}\right)^2}{df_s(\theta(g))^2} = 2\log \frac{\lambda_s^{(L)}}{df_s(\theta(g))},$$

$$\langle NIDF_i(g)\rangle - NIDF_i^{(L)}(g) = \log \frac{\left(\lambda_s^{(U)}\right)^2}{df_s(\theta(g))^2} = 2\log \frac{\lambda_s^{(U)}}{df_s(\theta(g))}.$$

Thus, the $NIDF_i(g)$ error is twice as large as those of $IDF(g)$ and $NIDF_d(g)$ if the observed document frequency in D_s is the same.

5.2.2. Approximation on Wavelet Tree. The subset-based approximation error depends solely on the document frequency in a subset corpus. This makes it difficult to control the error of the weights for all valid N-grams. A straightforward method to approximate the N-gram IDF weights for all N-grams is to prepare a single subset corpus $D_s \subset D$ and build a wavelet tree of D_s. However, the observed document frequency in D_s varies depending on the N-gram, which results in unbalanced and uncontrollable weight errors. In addition, many N-grams do not occur in D_s. The observed document frequency should be controllable to estimate the error of the weights, that is, a tailored subset corpus for each N-gram is required to observe the document frequency of a given number.

We propose an approximation method using the wavelet tree to tailor the subset corpus in which the document frequency equals a predefined number. The key idea is to stop the recursive function (Algorithm 1) when predefined document frequency df_p is observed. df_p corresponds to a sample from the Poisson distribution with parameter $\lambda_s = \langle df_s(g)\rangle$. Importantly, we count the number of documents that do not contain a search term, that is, negative document frequency $\overline{df_s}$, as well as document frequency df_s, and we incrementally construct subset corpus $D_s = \{0, 1, 2, \ldots, |D_s| - 1\}$ such that $|D_s| = df_s + \overline{df_s}$. Figure 6 illustrates the approximation method when $df_p = 2$.

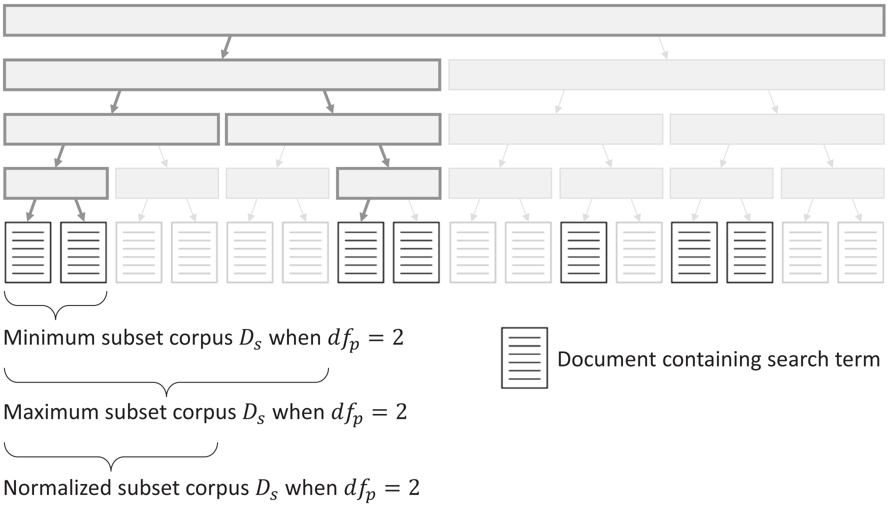

Minimum subset corpus D_s when $df_p = 2$
Maximum subset corpus D_s when $df_p = 2$
Normalized subset corpus D_s when $df_p = 2$
Document containing search term

Fig. 6. Example of the approximation method when $df_p = 2$. Traversing leftmost $df_p + 1$ (i.e., three) documents is sufficient to determine normalized subset corpus D_s among all possible D_s.

While there are multiple possible D_s containing df_p, normalized D_s can be obtained by traversing $df_p + 1$ documents. Our approximation method can be implemented with minor modifications to Algorithm 1. The time complexity for approximating $df(\theta(g))$ is roughly $O(N \cdot df_p \cdot \log \frac{|D|}{df_p})$. Here, df_p balances the tradeoff between time complexity and N-gram IDF weight errors.

We explain the modifications along with Algorithm 2, which describes a recursive function that counts df_s and $\overline{df_s}$ until df_s exceeds df_p. An additional initial input is predefined document frequency df_p. In addition, observed document frequency df_o is passed to child nodes in order to know when to stop the recursion. The first part of Algorithm 2, which calculates the next positions in child nodes (Lines 4–18), is the same as Algorithm 1, except that Algorithm 2 initializes $\overline{df_s}$. Visiting the child node when flag f_0 or f_1 is true (Lines 19–20, Lines 29–30) is similar but two arguments are increased.

The main difference between Algorithm 1 and Algorithm 2 is that $\overline{df_s}$ of a child node is counted when it is not visited, that is, f_0 or f_1 is false (Lines 21–27, Lines 31–37). If the wavelet tree is a perfect binary tree of height $\lceil \log |D| \rceil + 1$ (including leaf nodes), then the number of leaf nodes under a child node at depth $k + 1$ is equal to $2^{\lceil \log |D| \rceil - k}$. Practically, it cannot be a perfect binary tree when $\lceil \log |D| \rceil \neq \log |D|$. Therefore, we must cope with irregular nodes where the number of leaf nodes is less than $2^{\lceil \log |D| \rceil - k}$. Here, we assume incremental integers as document identifiers, that is, $D = \{0, 1, 2, \dots, |D| - 1\}$. Since the wavelet tree classifies each document into node 0 (left) or 1 (right) at depth $k + 1$ based on its kth bit, leaf nodes at depth $\lceil \log |D| \rceil + 1$ are created in order from left to right. The resultant wavelet tree has the property that all nodes, except the rightmost node (i.e., end position $p_e = n$), in every B_k are perfect binary trees. In addition, all leaf nodes are checked when the negative document frequency of the rightmost node is counted. This occurs when the document frequency in D is not greater than df_p. In this case, we can set $\overline{df_s}$ as $|D| - df_s$ (Lines 40–41).

Algorithm 2 stops the recursion when observed document frequency df_o plus additional document frequency df_s of child node 0 exceeds df_p, that is, it continues the

ALGORITHM 2: CountSubsetDF

```
Input: Start indices $i_s[0, m-1]$, end indices $i_e[0, m-1]$, numbers of occurrences
       $o[0, m-1]$, predefined document frequency $df_p$, observed document frequency
       $df_o = 0$, depth $k = 1$, start position $p_s = 0$, end position $p_e = n$
Output: Document frequency $df_s$, negative document frequency $\overline{df_s}$
if $k > \lceil \log |D| \rceil$ then // reach the leaf node, return ($df_s$, $\overline{df_s}$)
    return (1, 0);
end
$df_s = 0$, $\overline{df_s} = 0$, $f_0 =$ true, $f_1$ = true;
$p_b = p_s + rank_0(B_k, p_e) - rank_0(B_k, p_s)$;
init $i_{s0}[0, m-1]$, $i_{e0}[0, m-1]$, $i_{s1}[0, m-1]$, $i_{e1}[0, m-1]$;
for $j = 0$ to $m-1$ do
    $i_{s0}[j] = rank_0(B_k, p_s + i_s[j]) - rank_0(B_k, p_s)$;
    $i_{e0}[j] = rank_0(B_k, p_s + i_e[j]) - rank_0(B_k, p_s)$;
    $i_{s1}[j] = rank_1(B_k, p_s + i_s[j]) - rank_1(B_k, p_s)$;
    $i_{e1}[j] = rank_1(B_k, p_s + i_e[j]) - rank_1(B_k, p_s)$;
    if $i_{e0}[j] - i_{s0}[j] < o[j]$ then
        $f_0 =$ false;
    end
    if $i_{e1}[j] - i_{s1}[j] < o[j]$ then
        $f_1 =$ false;
    end
end
if $f_0$ then // visit child node 0
    $(df_s, \overline{df_s}) = CountSubsetDF(i_{s0}, i_{e0}, o, df_p, df_o, p_s, p_b, k+1)$;
else// count the negative document frequency of child node 0
    if $df_o == df_p$ then // get the median of minimum and maximum $D_s$
        $\overline{df_s} = 2^{\lceil \log |D| \rceil - k - 1}$;
    else
        $\overline{df_s} = 2^{\lceil \log |D| \rceil - k}$;
    end
end
if $df_o + df_s \leq df_p$ then // continue the recursion
    if $f_1$ then // visit child node 1
        $(df_s, \overline{df_s}) = (df_s, \overline{df_s}) + CountSubsetDF(i_{s1}, i_{e1}, o, df_p, df_o + df_s, p_b, p_e, k+1)$;
    else// count the negative document frequency of child node 1
        if $df_o + df_s == df_p$ then // get the median of minimum and maximum $D_s$
            $\overline{df_s} = \overline{df_s} + 2^{\lceil \log |D| \rceil - k - 1}$;
        else
            $\overline{df_s} = \overline{df_s} + 2^{\lceil \log |D| \rceil - k}$;
        end
    end
end
if $k == 1$ then // finalize
    if $df_s \leq df_p$ then // cope with irregular rightmost nodes
        $\overline{df_s} = |D| - df_s$;
    else// decrement $df_s$ since the recursion stops when observing $df_p + 1$
        $df_s = df_s - 1$;
    end
end
return $(df_s, \overline{df_s})$;
```

recursion when $df_o + df_s \leq df_p$ (Line 28). Recursion continues even when $df_o + df_s = df_p$ to normalize subset corpus size $|D_s|$. If the recursion is stopped as soon as $df_o + df_s$ reaches df_p, then $|D_s|$ becomes minimum (Figure 6) among all possible corpora in which the document frequency is df_p. Therefore, we take the average (or median) of D_s for all possible corpora to normalize $|D_s|$. Algorithm 2 counts half of the negative document frequency when the observed document frequency is equal to df_p (Lines 22–23, Lines 32–33). When $df_o + df_s$ reaches $df_p + 1$, it stops the recursion (Line 28) and decrements df_s to df_p (Lines 42–44).

Algorithm 2 assumes that the order of documents in corpus D is random. Randomizing the list of D and choosing an arbitrary number of leftmost documents to build $D_s = \{0, 1, 2, \ldots, |D_s| - 1\}$ gives equal opportunities for all possible subsets of the corpus, thereby ensuring that D_s is a random sample from a Poisson distribution. In practice, documents are usually aligned based on some aspects, for example, creation date. Thus, we introduce a random permutation of the list of D. We create random permutation list $L_R[0, |D| - 1]$ such that $L_R[i]$ for document identifier i has unique random identifier j $(0 \leq j < |D|)$. The Fisher-Yates shuffle [Durstenfeld 1964] can be adopted to create L_R. Each document identifier i in L_D is replaced with $L_R[i]$, and then the wavelet tree is constructed for L_D.

6. EXPERIMENTS

We evaluated the robustness of N-gram IDF using several applications. Specifically, we conducted experiments on key term extraction, web search query segmentation, ad hoc information retrieval, and text classification tasks. Key term extraction and web search query segmentation require selection of meaningful N-grams among all possible N-grams in the given texts. We successfully verified the capability of detecting meaningful words and phrases using dominant N-grams. On the other hand, we obtained negative results with ad hoc information retrieval and text classification tasks where the weights of the N-grams were directly used. We also evaluated our approximate computation method in terms of processing time, error distribution, and influence on the performance of key term extraction and web search query segmentation tasks. The universality of N-gram IDF was also tested by applying it to various languages other than English (Spanish, German, French, and Japanese).

6.1. Implementation and Processing

We first describe the implementation and processing used throughout the experimental section. We implemented the exact and approximate computation methods described in Section 5 in C++ using esaxx,[7] a library for building an enhanced suffix array, and wat-array,[8] a library for constructing a wavelet tree. The source code can be found on the web.[9] We processed English Wikipedia dump data from October 1, 2013 (approximately 11GB), using the implementation. We set threshold $\delta = 5$ for the minimum frequency of valid N-grams and the maximum N to 10 to reduce processing time.[10] We assumed that most meaningful N-grams in the corpus satisfied the above very loose constraints, and thus they did not negatively influence performance. Note that we omitted punctuation characters and capitalization information. In the exact method, we had to introduce a heuristic technique to complete processing. Specifically, the dynamic threshold of the document frequency, that is, $\frac{1}{2000}$ of the document frequency of each constituent word, was used in addition to δ. We processed Spanish (November 29, 2014), German

[7] https://code.google.com/p/esaxx/.

[8] https://code.google.com/p/wat-array/.

[9] http://iwnsew.com/.

[10] The influence of δ and the maximum N on performance and processing time is examined in Section 6.2.3.

(September 1, 2015), French (September 1, 2015), and Japanese (November 22, 2014) Wikipedia dump data. The same settings ($\delta = 5$, $N \leq 10$) were applied to these languages, except Japanese, because the characteristics of this language differ significantly from those of English, for example, Japanese has a large number of unique characters and does not employ word delimiters. We performed word segmentation using MeCab [Kudo et al. 2004] (version 0.996) and set the maximum N to 15 for Japanese.

6.2. Key Term Extraction

We evaluated the performance of exact N-gram IDF weights on the key term extraction task using the English Wikipedia dump (October 1, 2013). In Wikipedia, key terms are highlighted as anchor texts linked to their respective Wikipedia articles. Thus, we used anchor texts and emphasized texts (bold) as correct key terms. Given a Wikipedia article, each method extracted terms from the text, calculated the weight for the terms, and created a ranked list of key terms based on the weight. The key term extraction task consists of two sub tasks.

(1) Term detection: detect terms (key term candidates) among all N-grams
(2) Scoring: compute the score for key term candidates to rank them

We evaluated N-gram IDF from both aspects.

As a performance metric, we measured R-Prec, which is the precision of the top R terms for R correct key terms. To reduce the effect of local term weighting schemes, we built a dataset using the first paragraph of randomly chosen articles. In short texts, $TF = 1$ *challenge* [Timonen 2013] exists, that is, local term weight TF becomes 1 in most cases. Therefore, we can look into the performance of global term weighting schemes using short text datasets. The created dataset contained 1,678 short texts, in which there were 60.2 words (max: 291, min: 8) and 6.7 key terms[11] (max: 30, min: 3) on average. The created dataset can be found on the web.

6.2.1. Unsupervised Manner. Our N-gram IDF scheme can extract key term candidates and rank them based on only the weight. Dominant N-grams in a text are the candidates of the key term. A combination of TF[12] and N-gram IDF (hereafter TF-NIDF) was employed as the scoring method. As a comparative method, we employed a Part-of-Speech (POS) tagging-based method (noun phrase-based method), which has been proven very robust among representative unsupervised key term extraction methods, including TextRank [Mihalcea and Tarau 2004], SingleRank [Wan and Xiao 2008], ExpandRank [Wan and Xiao 2008], and KeyCluster [Liu et al. 2009], in the literature [Hasan and Ng 2010]. It first enumerates all noun phrases (NP) by detecting consecutive noun (and adjective) words, that is, Penn Treebank tags NN (noun, singular or mass), NNS (noun, plural), NNP (proper noun, singular), NNPS (proper noun, plural), and JJ (adjective) [Hasan and Ng 2010; Wan and Xiao 2008]. Here, we used the Stanford POS Tagger [Toutanova et al. 2003] (version 3.5.2, english-left3words-distsim.tagger model) to obtain POS tags. Then, it measures the weights of noun phrases by summing the TF-IDF weights of all constituent words [Robertson 2004]. We refer to this as TF-IDF sum. We also compared some alternative scoring methods, including TF-NIDF$_d$ (direct N-gram IDF), TF-IDF avg (average IDF of all constituent words), and TF-IDF (IDF of the phrase itself), as well as TF-NIDF and TF-IDF sum. These scoring methods were utilized with dominant N-grams or the noun phrase-based method.

Other comparative methods included the PMI-based method, informativeness [Tomokiyo and Hurst 2003], and TF-IDF unigram. The PMI-based method measures

[11] We omitted short texts with fewer than three key terms.

[12] Note that TF barely affected performance, because it was mostly 1 in our short text datasets.

Table III. Performance of Unsupervised Key Term Extraction on English Wikipedia First Paragraph Dataset

Term detection method	Scoring method	R-Prec
Dominant N-gram	TF-NIDF	$^{\circ\circ}_{\diamond}$0.386
Dominant N-gram	TF-NIDF$_d$	0.362
Dominant N-gram	TF-IDF sum	0.362
Dominant N-gram	TF-IDF avg	0.359
Dominant N-gram	TF-IDF	0.310
Dominant N-gram	Informativeness	0.302
Dominant N-gram (removing phrases including stop words)	TF-NIDF	$_{\diamond\diamond}$0.403
Dominant N-gram (removing phrases including stop words)	TF-NIDF$_d$	0.396
Dominant N-gram (removing phrases including stop words)	TF-IDF sum	0.404
Dominant N-gram (removing phrases including stop words)	TF-IDF avg	0.367
Dominant N-gram (removing phrases including stop words)	TF-IDF	0.360
Dominant N-gram (removing phrases including stop words)	Informativeness	0.345
Noun phrase	TF-NIDF	0.426
Noun phrase	TF-NIDF$_d$	0.425
Noun phrase	TF-IDF sum	0.428
Noun phrase	TF-IDF avg	0.409
Noun phrase	TF-IDF	0.410
Noun phrase	Informativeness	0.400
Noun phrase (no capitalization information)	TF-NIDF	0.405
Noun phrase (no capitalization information)	TF-NIDF$_d$	0.405
Noun phrase (no capitalization information)	TF-IDF sum	0.405
Noun phrase (no capitalization information)	TF-IDF avg	0.388
Noun phrase (no capitalization information)	TF-IDF	0.390
Noun phrase (no capitalization information)	Informativeness	0.381
No method (removing phrases including stop words)	TF-NIDF	$^{\circ\circ}$0.341
No method (removing phrases including stop words)	TF-NIDF$_d$	0.320
No method (removing phrases including stop words)	TF-IDF sum	0.324
No method (removing phrases including stop words)	TF-IDF avg	0.290
No method (removing phrases including stop words)	TF-IDF	0.252
No method (removing phrases including stop words)	Informativeness	0.244
PMI (best threshold 5.9, removing phrases including stop words)	TF-IDF sum	0.375
Unigram (removing stop words)	TF-IDF	0.229

Note: upper "○○" means two-sided t-test ($p < 0.01$) compared to TF-IDF sum with the same term detection method; lower "◇◇" and "◇" mean two-sided t-test ($p < 0.01$ and $p < 0.05$, respectively) compared to TF-IDF sum with PMI.

the arithmetic average PMI score for multiword phrases and detects MWEs rather than constituent words when the PMI score exceeds a threshold. Then, the score of single words and MWEs is measured using TF-IDF sum. Note that informativeness does not have a term detection phase. It directly measures the score of all N-grams using the relative probability of N-grams in the target text and background corpus (i.e., all of Wikipedia). We used informativeness with several term detection methods. Note that TF-IDF unigram only focuses on single words.

Table III shows the performance results of unsupervised key term extraction.[13] TF-IDF sum with noun phrases [Hasan and Ng 2010] achieved the best precision (0.428). N-gram IDF (dominant N-grams and TF-NIDF) scored 0.386 or 0.403 if phrases containing stop words were discarded. In other words, the dominant N-gram-based method showed somewhat worse performance than the noun-phrase-based method, aside from

[13] The scores differ from those presented in our conference article [Shirakawa et al. 2015], because, in this article, we selected a state-of-the-art POS tagger and performed fine-tuning to remove irregular symbols.

Table IV. Comparison between N-gram IDF (Dominant N-gram) and MED for Unsupervised Key Term Extraction Task. TF-NIDF Was Used as the Scoring Method. Phrases with Stop Words Were Removed in All Methods

Term detection method	R-Prec
Dominant N-gram	$^{\circ\circ}$0.403
MED (threshold 1.0)	0.344
MED (threshold 2.0)	0.375
MED (threshold 3.0)	0.386
MED (threshold 4.0)	0.389
MED (threshold 5.0)	0.390
MED (threshold 6.0)	0.392
MED (threshold 7.0)	0.392
MED (threshold 8.0)	0.392

Note: upper "○○" means two-sided t-test ($p < 0.01$) compared to MED with any threshold.

the need of a POS Tagger. However, the precision of TF-IDF sum with noun phrases was reduced to 0.405 when capitalization information was omitted, because the accuracy of the obtained POS tags deteriorated due to missing capitalization. The accuracy of the POS tags directly affected the quality of noun phrases and thus the accuracy of key terms. Since the dominant N-gram-based method does not require POS tags, it has advantages in noisy or domain-specific short texts in which the accuracy of the POS tags tends to decrease.

Focusing on statistical methods that do not require POS tags, N-gram IDF outperformed the PMI-based method and TF-IDF with unigram. Given that the threshold of the PMI-based method was wel configured, we can say that N-gram IDF could better leverage statistical information without parameter adjustments. TF-IDF unigram was apparently worse than the other methods, because it discarded phrases and detected incorrect words (unigrams) that should have composed phrases.

Here, we discuss a comparison of the scoring methods. TF-NIDF was better than other methods, including TF-IDF sum, when the key term candidates contained many incomplete phrases (i.e., with dominant N-gram or no method as the term detection method). This demonstrated that TF-NIDF can better weight N-grams of any length than TF-IDF sum. However, TF-NIDF was not better than TF-IDF sum when the term detection methods were reasonably accurate, because the scoring methods do not need to include the compositionality of multiword N-grams into the score once they were detected as complete terms or units. In such cases, TF-IDF sum, which has been theoretically justified on the condition that each word occurrence is statistically independent [Robertson 2004], worked well to compute the weights of terms. We conclude that the advantage of N-gram IDF lies in the comparability of the weight among overlapping unigrams and multiword N-grams to detect terms rather than in the scoring function used to rank terms.

We also compared N-gram IDF to MED. In the MED-based method, N-grams whose MED was smaller than a predefined threshold were detected as key term candidates. We then ranked the key term candidates using the same scoring method, that is, TF-NIDF. When an N-gram was ranked higher than overlapping shorter N-grams, they were discarded. Table IV shows a comparison between the dominant N-gram-based method and MED for the unsupervised key term extraction task. We changed the MED threshold from 0.1 to 8.0 at intervals of 0.1 but only list representative results to eliminate redundancy. MED demonstrated the same maximum precision when the threshold was 7.0 or greater. However, it was still worse than TF-NIDF with dominant

Table V. Performance of Supervised Key Term Extraction on English Wikipedia First Paragraph Dataset

Features	R-Prec
Basic features	0.357
+PMI, single word feature	0.389
+MED, single word feature	0.396
+Direct N-gram IDF	0.399
+N-gram IDF	0.402
+Dominant N-gram feature	$^{\circ\circ}$0.448
+Dominant N-gram feature, PMI, single word feature	0.459
+Dominant N-gram feature, MED, single word feature	0.463
+Dominant N-gram feature, direct N-gram IDF	0.459
+Dominant N-gram feature, N-gram IDF	0.457

Note: upper "∘∘" means two-sided t-test ($p < 0.01$) compared to {PMI, single word feature}, {MED, single word feature}, Direct N-gram IDF, N-gram IDF, and the best unsupervised method in Table III.

N-grams. Term detection using the MED threshold deteriorated performance when the key term candidates were ranked by TF-NIDF weights. This indicates that N-gram IDF retains the advantage of MED within the term weighting scheme.

6.2.2. Supervised Manner. While N-gram IDF can be used directly to extract key terms from texts by determining dominant N-grams, we note that N-gram IDF is a general-purpose measure rather than a method for specific applications. In this section, we leverage N-gram IDF as the features of supervised key term extraction [Turney 2003; Witten et al. 1999]. Here, we adopted a soft margin Support Vector Machine (SVM) with the radial basis function (RBF) kernel [Cortes and Vapnik 1995] and used the classification probability [Wu et al. 2004] to rank key terms. We utilized the first paragraph of randomly chosen Wikipedia articles (not in the test data) as training data, that is, anchor texts and bold texts were used as positive labels and others as negative labels. In total, 1,249 positive labels and 6,234 negative labels[14] from 208 first paragraphs were obtained.

We set the basic features as follows: N-gram length, TF, global TF (total occurrences in a corpus), IDF, and stop word feature (whether a stop word occurs at the beginning or end of the N-gram). In addition, the following feature sets were combined:

(1) PMI (arithmetic average), single word feature (whether the N-gram is a word)
(2) MED, single word feature
(3) Direct N-gram IDF
(4) N-gram IDF
(5) Dominant N-gram feature (whether the N-gram is a dominant N-gram)
(6) Dominant N-gram feature, PMI, single word feature
(7) Dominant N-gram feature, MED, single word feature
(8) Dominant N-gram feature, direct N-gram IDF
(9) Dominant N-gram feature, N-gram IDF

The single word feature was introduced, because PMI and MED could not define scores for single words (set to 0.0). For each feature set, we trained an SVM classifier and set the best cost and gamma parameters from $\{1 \times 10^z, 2 \times 10^z, 5 \times 10^z\}$ (z is an integer) via grid search using fivefold cross validation.

The performance results of supervised key term extraction are shown in Table V. First, the supervised method with the dominant N-gram feature drastically improved

[14] We sampled negative labels, because they were far more than positive labels.

Table VI. Comparison of Parameter Settings on Unsupervised Key Term Extraction Task. δ and N Are the Minimum Frequency and Length of Valid N-grams, Respectively

Parameter settings	R-Prec
$\delta = 5$, $N \leq 10$	0.403
$\delta = 6$, $N \leq 10$	$^{\bullet\bullet}$0.400
$\delta = 7$, $N \leq 10$	$^{\bullet\bullet}$0.396
$\delta = 8$, $N \leq 10$	$^{\bullet\bullet}$0.392
$\delta = 9$, $N \leq 10$	$^{\bullet\bullet}$0.390
$\delta = 10$, $N \leq 10$	$^{\bullet\bullet}$0.387
$\delta = 15$, $N \leq 10$	$^{\bullet\bullet}$0.378
$\delta = 20$, $N \leq 10$	$^{\bullet\bullet}$0.369
$\delta = 5$, $N \leq 7$	0.403
$\delta = 5$, $N \leq 5$	0.403
$\delta = 5$, $N \leq 4$	0.403
$\delta = 5$, $N \leq 3$	0.402
$\delta = 5$, $N \leq 2$	$^{\bullet\bullet}$0.394

Note: upper "●●" means two-sided t-test ($p < 0.01$) compared to $\delta = 5$, $N \leq 10$.

performance. With the help of the training data, it even outperformed the robust noun-phrase-based method (Table III). It further improved performance using one of the absolute scores obtained with PMI, MED, or N-gram IDF. The N-gram IDF weight as a feature did not contribute to performance compared to the dominant N-gram feature, which reflects relative N-gram IDF weights among overlapping N-grams. We again conclude that the potential of N-gram IDF can be exploited fully by comparing weights among overlapping N-grams. In addition, significant performance improvements obtained using N-gram IDF as features are of great promise, because there are various applications that can potentially incorporate N-gram IDF as features.

6.2.3. Influence of Parameters on Performance and Processing Time. We examined the influence of minimum frequency δ and maximum N-gram length N on performance and processing time. We measured the performance of the unsupervised key term extraction task with varying δ (5, 6, 7, 8, 9, 10, 15, and 20) and maximum N (10, 7, 5, 4, 3, and 2). The computation of N-gram IDF with various parameter settings was difficult; therefore, we estimated processing time by sampling approximately $\frac{1}{500}$ of valid N-grams satisfying the δ and N thresholds. We used a high-memory (greater than 60 GB) machine (Intel(R) Xeon(R) Processor E5-2643 v2 @ 3.50 GHz) to measure processing time.

Table VI shows the performance of the unsupervised key term extraction task. Parameters $\delta = 5$ and $N \leq 10$ were the default settings, as stated in Section 6.1. First, the minimum frequency δ for valid N-grams tended to deteriorate performance. This indicates that some key terms were very rare in the entire corpus. Thus, δ should be chosen carefully to cover such rare key terms. Second, the maximum N could be reduced without sacrificing performance; $N \leq 4$ or $N \leq 3$ was sufficient to maintain performance with our key term extraction dataset. Note that this might not be true when the target domain contains longer key terms (e.g., the entertainment domain contains many long phrases, such as song and movie titles).

Figure 7 shows the estimated processing time to compute all valid N-grams satisfying the thresholds. Greater δ or smaller N decreased the estimated processing time to some extent. When we set $N \leq 3$ to maintain performance, it was reduced to between one-half and one-third of the estimated processing time with $N \leq 10$. However, its

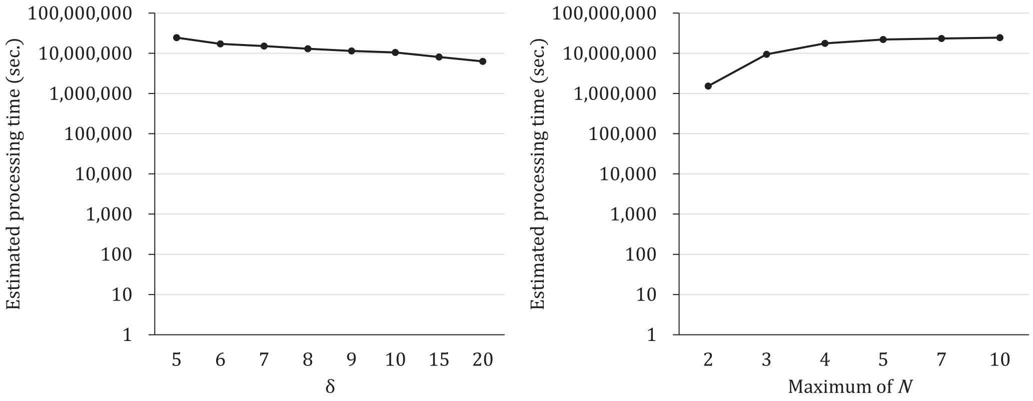


Fig. 7. Estimated processing time when changing δ and maximum N.

effect was limited, because it still required more than 100 days. Setting much greater δ and small N can solve the computation problem at the expense of performance.

6.3. Web Search Query Segmentation

Web search query segmentation is another application of N-gram IDF. In this evaluation, we used an information retrieval-based web search query segmentation dataset [Roy et al. 2012]. This dataset contains 13,959 web documents, 500 test queries, and their query-relevance sets (qrels). The qrels were created by three human experts grading the query-document relevance score as 2 (highly relevant), 1 (relevant), or 0 (irrelevant). The average relevance score of the three experts was used as the gold standard in our evaluation. Given a query (e.g., "larry the lawnmower tv show"), each method segmented it into sets of words and phrases (e.g., "larry the lawnmower" and "tv show"). The phrases were then used to force the search system to match them exactly in documents. This is a Boolean query using double quotes in general web search engines. After collecting documents that satisfy the Boolean query, we calculated the conventional TF-IDF weight for each document. The ranking (scoring) function in this evaluation was fixed to a conventional TF-IDF retrieval model. Here, IDF was computed using the dataset documents. Selecting which phrases should be quoted is a difficult problem; thus, Roy et al. [2012] measured evaluation scores for all possible quoted queries given a segmentation result obtained by each method. We followed their procedure in our evaluation.

In our method, we utilized dominant N-grams determined by N-gram IDF as the query segmentation results. N-gram IDF was computed using the dataset documents, because many queries and documents were informal and Wikipedia does not cover some N-grams in the dataset. Processing 283 MB of the dataset documents using our exact computation method took 40min. As a comparative method, we again adopted a POS tagging-based method. Since web search queries are written informally, it is difficult to find noun phrases by simply detecting consecutive NN* and JJ tags. Therefore, we utilized frequent POS patterns in lexicons reported in the literature [Roy et al. 2014]: NN NN, NN NN NN, JJ NN NN, JJ NN, NN NNS, NN NN NNS, NN IN NN, FW FW, JJ JJ NN, and JJ NN NNS (NNP and NNPS can be substituted for NN and NNS, respectively). A query with POS tags was segmented such that the number of phrases became minimum. We applied the left-longer approach to queries with multiple segmentation results (e.g., "NN NN NN and NN NN" and "NN NN and NN NN NN").

The dataset [Roy et al. 2012] also includes segmentation results obtained with other representative methods, that is, Mishra et al. [2011], Mishra et al. with Wiki,

Table VII. Performance of Query Segmentation on Roy et al. Dataset

Method	nDCG@5	nDCG@10	MAP@5	MAP@10	MRR@5	MRR@10
N-gram IDF (dominant N-gram)	${}_{\circ\circ}0.730$	${}^{\bullet}0.742$	0.900	${}^{\bullet}0.893$	${}^{\circ}0.582$	${}^{\circ}0.593$
POS tagging-based method	0.699	0.726	0.883	0.878	0.535	0.546
Mishra et al.	0.706	0.737	0.895	0.892	0.529	0.542
Mishra et al. with Wiki	0.725	0.750	0.907	0.902	0.561	0.571
PMI-Q	0.716	0.736	0.898	0.892	0.567	0.577
PMI-W	0.670	0.707	0.860	0.863	0.493	0.506
Unsegmented	0.655	0.689	0.852	0.854	0.465	0.481
Human A	0.728	0.746	0.904	0.899	0.575	0.585
Human B	0.727	0.747	0.903	0.898	0.567	0.577
Human C	0.717	0.744	0.899	0.896	0.543	0.555
BQV	0.765	0.768	0.927	0.914	0.673	0.680

Note: upper "○" and "•" mean two-sided t-test ($p < 0.05$) compared to Mishra et al. with Wiki; lower "○○" means two-sided t-test ($p < 0.01$) compared to PMI-Q.

PMI using query logs (PMI-Q) [Mishra et al. 2011], and PMI using web documents (PMI-W) [Hagen et al. 2011]. Mishra et al. used Hoeffding's inequality with query logs to obtain significant N-grams in a query. Mishra et al. with Wiki augmented the query segmentation results of Mishra et al. by utilizing Wikipedia titles. PMI was measured for any two consecutive words in a query and a segment break was inserted between them when the PMI was below a threshold. The PMI-Q and PMI-W thresholds were set to 0.156 and 8.141, respectively, in Roy et al. [2012]. In addition, the dataset includes unsegmented queries and (probably) the best quoted version of the queries (BQV).[15] The dataset also provides three segmentation results manually obtained by humans. Note that they differed from those who rated the qrels. We measured evaluation scores for segmentation results. We employed the same evaluation metrics: normalized Discounted Cumulative Gain (nDCG), Mean Average Precision (MAP), and Mean Reciprocal Rank (MRR) for the top five and 10 search results. Computing MAP and MRR requires binary values for the qrel score; thus, we considered 2 and 1 as relevant when computing MAP, and only 2 as relevant for MRR according to Roy et al.

Table VII shows the performance results.[16] N-gram IDF was competitive with well-adjusted methods, such as Mishra et al. with Wiki and PMI-Q, as well as the human segmentation results. Note that N-gram IDF only has the weight of each N-gram, similarly to conventional IDF. Mishra et al. with Wiki leverages human knowledge of Wikipedia and PMI-Q requires parameter tuning. Moreover, both are based on query logs, which usually cannot be used. N-gram IDF achieved competitive performance by simply selecting dominant N-grams based on the weight. These results demonstrate the robustness of N-gram IDF.

The POS tagging-based method, which was a strong baseline for key term extraction tasks, did not work well with the web search query segmentation task, because the robust rule for detecting noun phrases (i.e., consecutive NN* and JJ tags) could not be applied to the web search query, which was often a sequence of phrases rather than a sentence. Even when POS tag rules designed for web search queries [Roy et al. 2014] were introduced, the POS tagging-based method was less effective for query

[15] BQV in the literature [Roy et al. 2012] may not be the best, because it does not consider partly overlapping phrases, such as "free solitaire" and "solitaire card games" for the query "play free solitaire card games."

[16] Evaluation scores of the comparative methods differed from those reported by Roy et al. [2012], because we used the TF-IDF weight computed using the dataset documents. We observed tendencies that did not contradict their results.

Table VIII. Performance of Ad Hoc Information Retrieval on FIRE 2011 News Dataset

Method	MAP@5	MAP@10	MRR@5	MRR@10
TF-NIDF	0.494	0.473	0.667	0.678
TF-NIDF$_d$	0.481	0.464	0.630	0.645
TF-IDF sum	0.470	0.454	0.627	0.639
TF-IDF avg	0.504	0.480	0.682	0.692
TF-IDF	0.484	0.465	0.636	0.648
TF-IDF (unigram)	0.565	0.532	0.727	0.739

Table IX. Performance of Ad Hoc Information Retrieval on Roy et al. Dataset

Method	nDCG@5	nDCG@10	MAP@5	MAP@10	MRR@5	MRR@10
TF-NIDF	0.675	0.705	0.872	0.871	0.477	0.491
TF-NIDF$_d$	0.693	0.716	0.884	0.881	0.507	0.518
TF-IDF sum	0.696	0.720	0.886	0.882	0.513	0.523
TF-IDF avg	0.685	0.712	0.876	0.875	0.496	0.507
TF-IDF	0.695	0.718	0.885	0.881	0.508	0.520
TF-IDF (unigram)	0.655	0.689	0.852	0.854	0.465	0.481

segmentation than N-gram IDF. Thus, we consider that the effectiveness of N-gram IDF increases relatively with informal texts.

6.4. Negative Results: Ad Hoc Information Retrieval and Text Classification

We evaluated N-gram IDF with common applications of term weighting schemes, that is, ad hoc information retrieval and text classification. In both tasks, the weight of N-gram IDF was leveraged and evaluated with bag-of-words (specifically, bag-of-N-grams) representations. In the ad hoc information retrieval task, term weighting schemes were used to find the most relevant web pages given a query. A ranked list of web pages was generated based on term weights and evaluated. The evaluation metrics were nDCG (if graded qrel scores were available), MAP, and MRR for the top five and 10 search results. The text classification task utilized term weighting schemes for characterizing texts to classify them into predefined categories. We trained a linear SVM classifier and set the best cost parameter from $\{1 \times 10^z, 2 \times 10^z, 5 \times 10^z\}$ (z is an integer) using fivefold cross validation for each term weighting scheme. The micro-average precision and macro-average precision of classification results were measured as metrics for text classification. We used the following datasets:

(1) Ad hoc information retrieval: FIRE 2011 news dataset (English, title query)[17] and dataset [Roy et al. 2012]
(2) Text classification: 20 Newsgroups dataset (20news-bydate) [Lang 1995] and short text dataset [Phan et al. 2008].

We compared TF-NIDF, TF-NIDF$_d$, TF-IDF sum, TF-IDF avg, and TF-IDF with bag-of-N-grams representations.

Tables VIII, IX, X, and XI show the performance results on the FIRE 2011 news dataset, Roy et al. dataset, 20 Newsgroups dataset, and Phan et al. short text dataset, respectively. Overall, N-gram IDF (TF-NIDF or TF-NIDF$_d$) was not effective for both tasks. With the FIRE 2011 news dataset, the use of N-grams (regardless of the term weighting scheme) was counterproductive. In other words, unigrams were sufficient

[17] http://fire.irsi.res.in/fire/static/data.

Table X. Performance of Text Classification on 20 Newsgroups Dataset

Method	Micro	Macro
TF-NIDF	0.855	0.848
TF-NIDF$_d$	0.856	0.848
TF-IDF sum	0.853	0.846
TF-IDF avg	0.858	0.851
TF-IDF	0.851	0.844
TF-IDF (unigram)	0.856	0.848

Table XI. Performance of Text Classification on Phan et al. Short Text Dataset

Method	Micro	Macro
TF-NIDF	0.720	0.708
TF-NIDF$_d$	0.706	0.686
TF-IDF sum	0.696	0.677
TF-IDF avg	0.712	0.693
TF-IDF	0.702	0.682
TF-IDF (unigram)	0.723	0.711

and suitable to retrieve relevant web news pages in this dataset. The Roy et al. dataset, which was also used in the experiment on web search query segmentation (Section 6.3), contained complicated long queries and thus N-grams were helpful. However, N-gram IDF did not need to be used as the term weighting scheme for N-grams. Merely emphasizing longer N-grams (TF-IDF sum and TF-IDF) was better than moderately weighting N-grams (TF-NIDF and TF-IDF avg) to make the top five or 10 search results more relevant to the queries, which contrasts the results obtained with the FIRE 2011 dataset. Bringing together the results in Section 6.3, query segmentation appears to be more suitable than N-gram weighting when N-grams are useful in ad hoc information retrieval. In addition, N-grams were nearly meaningless with both datasets for the text classification task. Using only unigrams with the adjusted classifier was sufficient for common category-level classification.

6.5. Evaluation of Approximate N-gram IDF

We scrutinized our approximate computation method for estimating N-gram IDF weights relative to processing time, error distribution, and performance at an application level. Based on the information presented in this section, single parameter df_p can be determined to balance processing time and the allowable weight error.

6.5.1. Processing Time. We examined the processing time of the exact and approximate computation methods against corpus size. We constructed subset corpora of different sizes from the complete English Wikipedia corpus. Specifically, roughly $\frac{1}{1000}$, $\frac{1}{100}$, and $\frac{1}{10}$ of all articles (4,379,810 articles) were extracted randomly. We denote these as Subset1/1000, Subset1/100, and Subset1/10. We measured the processing time of the approximate method when df_p was 5, 10, 20, 50, and 100, as well as that of the exact method. Each of the methods is denoted Approx5, Approx10, Approx20, Approx50, and Approx100, respectively.

Figure 8 shows the processing time required to compute N-gram IDF weights for all valid N-grams. The numbers of valid N-grams were 108,261 (Subset1/1000), 920,378 (Subset1/100), 8,694,915 (Subset1/10), and 87,491,762 (Whole). The processing time of the exact method increased by $O(|D|^2)$ for corpus size $|D|$. We could not complete the exact computation process for the whole corpus without introducing the dynamic threshold. It still took 12 days using two high-memory (greater than 60GB) machines (Intel(R) Xeon(R) Processor E5-2643 v2 @ 3.50 GHz) with the dynamic threshold to obtain the weights of 39,610,779 N-grams. The estimated time of the exact method for processing the whole corpus is approximately 1 year.

The approximate methods successfully reduced processing time. It was reduced drastically when the corpus size was large. When $df_p = 10$ (Approx10), the computation of N-gram IDF with the whole corpus was completed within a day using a single high-memory machine. Even for $df_p = 100$ (Approx100), the total time was less than 5 days. Our approximation methods could process all of English Wikipedia in realistic time.

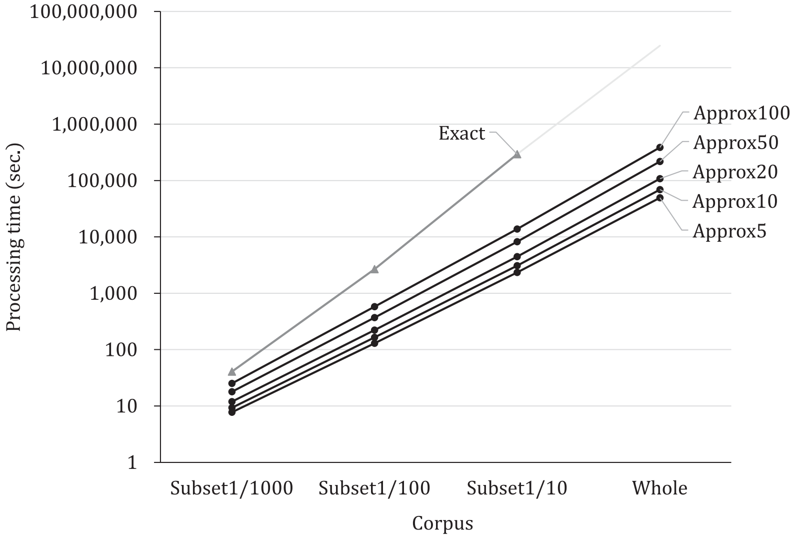
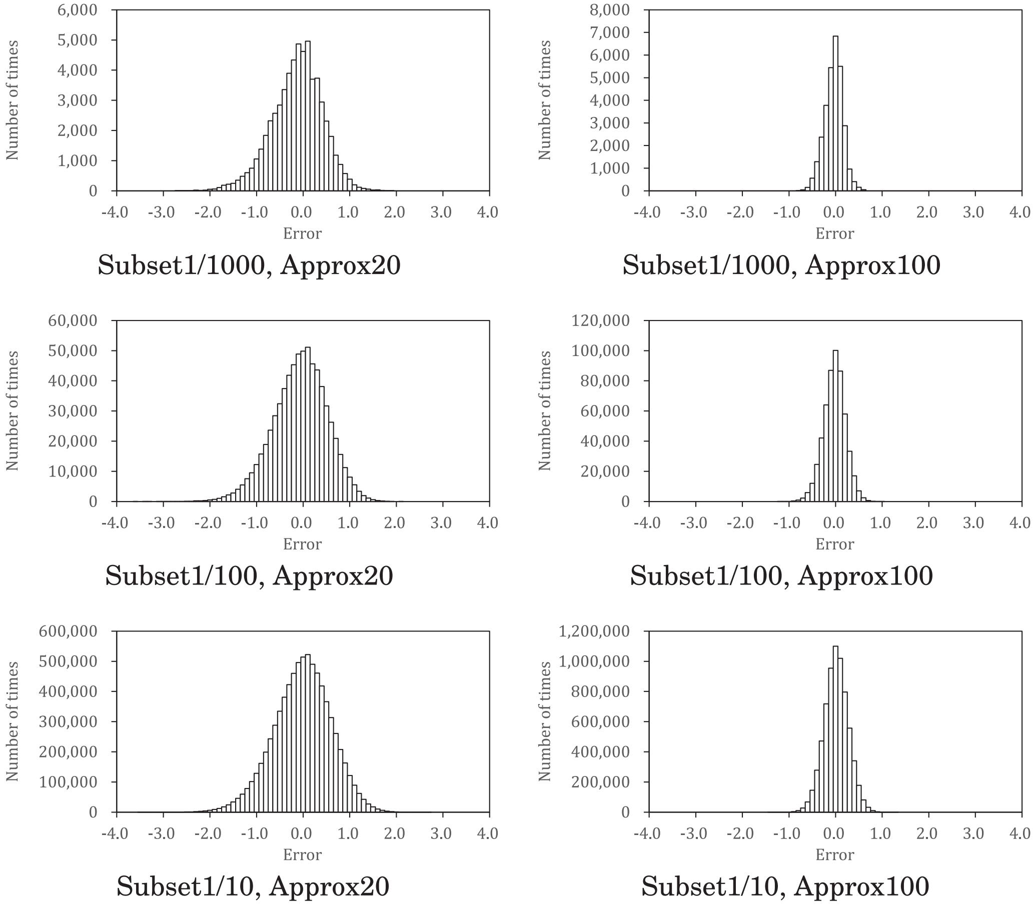


Fig. 8. Processing time of exact and approximate computation methods.

Our methods can be parallelized easily; therefore, one can benefit from processing speed-up in parallel computing situations.

The time complexity of the approximate methods was roughly $O(|D| \cdot df_p \cdot \log \frac{|D|}{df_p})$. In fact, the processing time increased slightly more than expected for corpus size $|D|$. The major reason for this was that $df(\theta(g))$ tended to be less than df_p with small corpora. When $df(\theta(g)) \leq df_p$, the number of times a leaf node was visited on a wavelet tree was $df(\theta(g))$ rather than $df_p + 1$; thus, the weight was obtained in less time. Small corpora contained a relatively large number of N-grams satisfying $df(\theta(g)) \leq df_p$ against all valid N-grams, for example, the rate of $df(\theta(g)) \leq 100$ was 72% in the Subset1/1000 corpus but only 11% in the whole corpus. Another reason was that the rate of long N-grams increased as corpus size increased. The average lengths of valid N-grams were 1.87 (Subset1/1000), 2.39 (Subset1/100), 2.93 (Subset1/10), and 3.50 (Whole). Since the time complexity for obtaining single N-gram IDF weight is roughly $O(N \cdot df_p \cdot \log \frac{|D|}{df_p})$, the average of N should affect total processing time.

6.5.2. Error Distribution. We investigated the error distribution of the approximate N-gram IDF weights obtained (Section 6.5.1) to demonstrate the controllability of the error of the weights. Figure 9 shows histograms of the error between the exact and approximate N-gram IDF weights in several environments. The weights of Approx20 ($df_p = 20$) and Approx100 ($df_p = 100$) were compared to the exact weights of the Subset1/1000, Subset1/100, and Subset1/10 corpora. Here, we omitted N-grams satisfying $df(\theta(g)) \leq df_p$, because their weights were computed exactly even with the approximate method. Figure 9 shows that the error distribution was determined by predefined (thus observed) document frequency df_p and not by corpus size $|D|$, as we showed analytically in Section 5.2.1. When $df_p = 20$ (Approx20), the underestimated and overestimated errors of N-gram IDF were $2 \log \frac{10.35}{20} \approx -1.90$ and $2 \log \frac{34.67}{20} \approx 1.58$, respectively, with 99% confidence. They became $2 \log \frac{76.12}{100} \approx -0.79$ and $2 \log \frac{128.76}{100} \approx 0.73$ when $df_p = 100$ (Approx100). The histograms in Figure 9 largely demonstrate the above values. Note that the estimated weight for each N-gram was practically not independent, because the same words appeared in many N-grams. This influence, however, turned out to



Subset1/1000, Approx20

Subset1/1000, Approx100

Subset1/100, Approx20

Subset1/100, Approx100

Subset1/10, Approx20

Subset1/10, Approx100

Fig. 9. Histograms of the error between the exact and approximate N-gram IDF weights.

be extremely limited. The above results demonstrate that the error of N-gram IDF weights was controllable by single parameter df_p regardless of corpus size.

Focusing on the edges of the error distribution in Figure 9, the maximum absolute errors tended to increase with corpus size or, more accurately, the number of valid N-grams. They were 2.686 (Subset1/1000), 3.649 (Subset1/100), and 3.533 (Subset1/10) for Approx20, and 0.793 (Subset1/1000), 1.205 (Subset1/100), and 1.437 (Subset1/10) for Approx100. While most errors fell into the confidence interval of the Poisson distribution, very few errors deviated significantly from the confidence interval. Such cases are inevitable in our approximation method. Nevertheless, this can be predictable and controllable to some degree. Given the number of valid N-grams, the probability that the maximum absolute error is less than a certain value can be estimated using Poisson distribution. Based on this probability, we can select df_p or run the whole computation several times with different random permutation lists of documents to diminish the maximum absolute error.

6.5.3. Application Performance. How the error of the N-gram IDF weight affects application performance should be clarified. Thus, we compared performance results between exact and approximate N-gram IDF weights. We conducted the same experiments for the key term extraction (Section 6.2) and web search query segmentation tasks

Table XII. Performance of Approximate N-gram IDF Weights on Key Term Extraction Task. Dominant N-gram and TF-NIDF Were Used as the Term Detection Method and Scoring Method, Respectively

Method	R-Prec
Approx N-gram IDF, $df_p = 5$	$^{\bullet\bullet}$0.358
Approx N-gram IDF, $df_p = 10$	$^{\bullet\bullet}$0.367
Approx N-gram IDF, $df_p = 20$	$^{\bullet\bullet}$0.376
Approx N-gram IDF, $df_p = 50$	0.382
Approx N-gram IDF, $df_p = 100$	0.384
Approx N-gram IDF, $df_p = 5$ (removing phrases including stop words)	$^{\bullet\bullet}$0.379
Approx N-gram IDF, $df_p = 10$ (removing phrases including stop words)	$^{\bullet\bullet}$0.387
Approx N-gram IDF, $df_p = 20$ (removing phrases including stop words)	$^{\bullet\bullet}$0.394
Approx N-gram IDF, $df_p = 50$ (removing phrases including stop words)	0.402
Approx N-gram IDF, $df_p = 100$ (removing phrases including stop words)	0.406
N-gram IDF	0.386
N-gram IDF (removing phrases including stop words)	0.403

Note: upper "••" means two-sided t-test ($p < 0.01$) compared to N-gram IDF (with the same settings for stop word-based removal).

Table XIII. Performance of Approximate N-gram IDF Weights on Web Search Query Segmentation Task. Dominant N-gram Was Used for Query Segmentation

Method	nDCG@5	nDCG@10	MAP@5	MAP@10	MRR@5	MRR@10
Approx N-gram IDF, $df_p = 5$	$^{\bullet}$0.725	$^{\bullet}$0.739	0.899	0.892	0.580	0.590
Approx N-gram IDF, $df_p = 10$	0.728	0.741	0.899	0.892	0.577	0.587
Approx N-gram IDF, $df_p = 20$	0.730	0.742	0.901	0.894	0.583	0.592
Approx N-gram IDF, $df_p = 50$	0.730	0.743	0.900	0.893	0.587	0.596
Approx N-gram IDF, $df_p = 100$	0.729	0.741	0.899	0.893	0.580	0.589
N-gram IDF	0.730	0.742	0.900	0.893	0.582	0.593

Note: upper "•" means two-sided t-test ($p < 0.05$) compared to N-gram IDF.

(Section 6.3). Tables XII and XIII show the performance of approximate N-gram IDF weights for both applications. With the key term extraction task, the precision of approximate weights was nearly equal to that of the exact weights when df_p was 50 or 100. The best precision was 0.406, which was achieved by approximate weights with $df_p = 100$, not by the exact weights. This was partly because the exact method had to skip computation of some valid N-grams using the dynamic threshold, which might discard potentially important N-grams. For example, "anglo american playing cards" was discarded by the dynamic threshold but was a key term. The approximate method calculated the weights for all valid N-grams and thus this key term was detected successfully. When df_p was 5, 10, or 20, precision decreased somewhat.

Approximate weights also achieved nearly the same performance with the web search query segmentation task (Table XIII). Differing from key term extraction, the performance of the query segmentation was fairly good even with small df_p, because the queries were likely to comprise multiple noncontiguous phrases. Finding joints between phrases was easier than detecting key terms from natural sentences. In addition, each segment did not need to be a complete term. In other words, detecting units that refined search results was sufficient to improve performance. The approximate weights with small df_p could detect such units from web search queries.

6.6. Performance on Different Languages

Finally, we tested the universality of N-gram IDF across languages. Specifically, we created key term extraction datasets for Spanish, German, French, and Japanese using

Table XIV. Statistics of Wikipedia First Paragraph Datasets

Language	Number of texts	Average number of words	Average number of key terms
English	1,678	60.2 (max: 291, min: 8)	6.7 (max: 30, min: 3)
Spanish	1,766	50.8 (max: 299, min: 7)	7.0 (max: 26, min: 3)
German	1,828	35.7 (max: 181, min: 8)	6.4 (max: 20, min: 3)
French	1,717	41.2 (max: 225, min: 6)	5.9 (max: 39, min: 3)
Japanese	1,890	56.8 (max: 379, min: 8)	7.5 (max: 24, min: 3)

Table XV. Performance of Key Term Extraction on Spanish Wikipedia First Paragraph Dataset. Phrases with Stop Words Were Removed in All Methods

Term detection method	Scoring method	R-Prec
Dominant N-gram	Approx TF-NIDF, $df_p = 5$	$^{\bullet\bullet}0.362$
Dominant N-gram	Approx TF-NIDF, $df_p = 10$	$^{\bullet\bullet}0.368$
Dominant N-gram	Approx TF-NIDF, $df_p = 20$	$^{\bullet\bullet}0.369$
Dominant N-gram	Approx TF-NIDF, $df_p = 50$	$^{\bullet\bullet}_{\circ\circ}0.376$
Dominant N-gram	Approx TF-NIDF, $df_p = 100$	$^{\bullet\bullet}_{\circ\circ}0.376$
Noun phrase	TF-IDF sum	0.398
Noun phrase (no capitalization information)	TF-IDF sum	0.357
PMI (best threshold 10.7)	TF-IDF sum	0.363
Unigram	TF-IDF	0.283

Note: upper “$\bullet\bullet$” means two-sided t-test ($p < 0.01$) compared to TF-IDF sum with noun phrases; lower “$\circ\circ$” means two-sided t-test ($p < 0.01$) compared to TF-IDF sum with PMI.

Table XVI. Performance of Key Term Extraction on German Wikipedia First Paragraph Dataset. Phrases with Stop Words Were Removed in All Methods

Term detection method	Scoring method	R-Prec
Dominant N-gram	Approx TF-NIDF, $df_p = 5$	0.437
Dominant N-gram	Approx TF-NIDF, $df_p = 10$	0.442
Dominant N-gram	Approx TF-NIDF, $df_p = 20$	0.447
Dominant N-gram	Approx TF-NIDF, $df_p = 50$	$_{\circ\circ}0.452$
Dominant N-gram	Approx TF-NIDF, $df_p = 100$	$_{\circ\circ}0.452$
Noun phrase	TF-IDF sum	0.445
Noun phrase (no capitalization information)	TF-IDF sum	0.370
PMI (best threshold 10.2)	TF-IDF sum	0.441
Unigram	TF-IDF	0.360

Note: lower “$\circ\circ$” means two-sided t-test ($p < 0.01$) compared to TF-IDF sum with PMI.

Wikipedia. We used the same dump data described in Section 6.1. With the exception of Japanese, we employed the Stanford POS Tagger (version 3.5.2) to obtain POS tags and detect NP. The models used were spanish-distsim.tagger (Spanish), german-hgc.tagger (German), and french.tagger (French). MeCab (version 0.996) was employed as the Japanese POS tagger to obtain word boundaries and POS tags. We applied the same rule to detect noun phrases, that is, consecutive nouns and adjectives. Table XIV shows the statistics of the created datasets and the English dataset. Due to the tremendous computational cost of exact computation, we only evaluated approximate N-gram IDF weights.

Tables XV, XVI, XVII, and XVIII show performance with the Spanish, German, French, and Japanese datasets, respectively. N-gram IDF worked reasonably well for all languages. When df_p was 50 or 100, there were significant performance improvements compared to TF-IDF sum with PMI even with the best thresholds. Given that both N-gram IDF and TF-IDF sum with PMI were based on the corpus-based statistical information, N-gram IDF turned out to better leverage the statistical information

Table XVII. Performance of Key Term Extraction on French Wikipedia First Paragraph Dataset. Phrases with Stop Words Were Removed in All Methods

Term detection method	Scoring method	R-Prec
Dominant N-gram	Approx TF-NIDF, $df_p = 5$	$^{\bullet\bullet}0.396$
Dominant N-gram	Approx TF-NIDF, $df_p = 10$	0.405
Dominant N-gram	Approx TF-NIDF, $df_p = 20$	0.411
Dominant N-gram	Approx TF-NIDF, $df_p = 50$	$_{\circ\circ}0.416$
Dominant N-gram	Approx TF-NIDF, $df_p = 100$	$_{\circ\circ}0.417$
Noun phrase	TF-IDF sum	0.412
Noun phrase (no capitalization information)	TF-IDF sum	0.374
PMI (best threshold 9.3)	TF-IDF sum	0.404
Unigram	TF-IDF	0.282

Note: upper "●●" means two-sided t-test ($p < 0.01$) compared to TF-IDF sum with noun phrases; lower "○○" means two-sided t-test ($p < 0.01$) compared to TF-IDF sum with PMI.

Table XVIII. Performance of Key Term Extraction on Japanese Wikipedia First Paragraph Dataset. Phrases with Stop Words Were Removed in All Methods

Term detection method	Scoring method	R-Prec
Dominant N-gram	Approx TF-NIDF, $df_p = 5$	$^{\bullet\bullet}_{\circ\circ}0.380$
Dominant N-gram	Approx TF-NIDF, $df_p = 10$	$^{\bullet\bullet}_{\circ\circ}0.396$
Dominant N-gram	Approx TF-NIDF, $df_p = 20$	$^{\bullet\bullet}_{\circ\circ}0.398$
Dominant N-gram	Approx TF-NIDF, $df_p = 50$	$^{\bullet\bullet}_{\circ\circ}0.400$
Dominant N-gram	Approx TF-NIDF, $df_p = 100$	$^{\bullet\bullet}_{\circ\circ}0.404$
Noun phrase	TF-IDF sum	0.436
PMI (best threshold 7.2)	TF-IDF sum	0.344
Unigram	TF-IDF	0.217

Note: upper "●●" means two-sided t-test ($p < 0.01$) compared to TF-IDF sum with noun phrases; lower "○○" means two-sided t-test ($p < 0.01$) compared to TF-IDF sum with PMI.

without relying on threshold adjustments. These results demonstrate the universality of N-gram IDF across languages.

The performance differences between N-gram IDF and TF-IDF sum with noun phrases [Hasan and Ng 2010] fluctuated depending on the language. For German and French, N-gram IDF was competitive with TF-IDF sum with noun phrases, even when the text was formal, that is, POS tagging results were reliable. With Spanish, the performance of N-gram IDF was better than that of TF-IDF sum with noun phrases only when capitalization information was missing. With Japanese, N-gram IDF was worse than TF-IDF sum with noun phrases. The differences seemed to arise from the characteristics of the language, for example, the ratio of key words and key phrases, and the rate of key terms consisting of successive nouns and adjectives. Since N-gram IDF only requires word segmentations (it does not require POS tags), the results are promising for language-independent text analysis.

7. CONCLUSIONS

This article revealed the relationship between IDF and information distance. Specifically, the IDF of a term is equal to the distance between the term and the empty string in the information distance space where the Kolmogorov complexity is approximated using web documents and Shannon-Fano coding. Our findings are helpful to design a new term weighting scheme, because the information distance can be interpreted as the term weight. Based on our findings, we have proposed a global term weighting scheme, that is, N-gram IDF, by incorporating IDF and MED, a universal information distance-based metric for measuring word compositionality. N-gram IDF enables us to compare the weights of N-grams for any *N* and thus determine dominant N-grams among overlapping N-grams using the magnitude relation of the weight. We have verified the robustness of N-gram IDF on unsupervised/supervised key term extraction and web search query segmentation tasks. N-gram IDF achieved promising results compared to state-of-the-art methods without using extra resources.

The computation of N-gram IDF weights is provably time consuming; therefore, we also proposed a fast approximation method based on the wavelet tree. Our method efficiently tailors a subset corpus in which document frequency is the same as a given number. With the help of Poisson distribution, the error of the approximate N-gram IDF weights can be estimated analytically from the document frequency in a subset corpus. The performance of approximate weights with applications was nearly the same as that of exact weights, while processing time was reduced drastically. In addition, the universality of N-gram IDF across languages was demonstrated with Spanish, German, French, and Japanese key term extraction tasks.

In future, we will handle character N-grams for completely language-independent N-gram weighting schemes. In this work, we were required to apply word segmentation to languages without spaces between words, such as Japanese. We will pursue the possibility of character-level N-gram IDF based on Kolmogorov complexity and Shannon's information theory, both of which are closely related [Li and Vitányi 2009].

REFERENCES

Mohamed Ibrahim Abouelhoda, Stefan Kurtz, and Enno Ohlebusch. 2004. Replacing suffix trees with enhanced suffix arrays. *J. Discr. Alg.* 2, 1 (Mar. 2004), 53–86.

Akiko Aizawa. 2003. An information-theoretic perspective of TF-IDF measures. *Inf. Process. Manag.* 39, 1 (Jan. 2003), 45–65.

Jérémy Barbay and Claire Kenyon. 2002. Adaptive intersection and t-threshold problems. In *Proceedings of ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 390–399.

Charles H. Bennett, Péter Gács, Ming Li, Paul M. B. Vitányi, and Wojciech H. Zurek. 1998. Information distance. *IEEE Trans. Inf. Theory* 44, 4 (July 1998), 1407–1423.

Anselm Blumer, J. Blumer, David Haussler, Ross M. McConnell, and Andrzej Ehrenfeucht. 1987. Complete inverted files for efficient text retrieval and analysis. *J. ACM* 34, 3 (July 1987), 578–595.

Abraham Bookstein and Don R. Swanson. 1974. Probabilistic models for automatic indexing. *J. Am. Soc. Inf. Sci.* 25, 5 (Sept./Oct. 1974), 312–318.

Gerlof Bouma. 2009. Normalized (pointwise) mutual information in collocation extraction. In *Proceedings of International Conference of the German Society for Computational Linguistics and Language Technology (GSCL)*. 31–40.

Fan Bu, Xiaoyan Zhu, and Ming Li. 2010. Measuring the non-compositionality of multiword expressions. In *Proceedings of International Conference on Computational Linguistics (COLING)*. 116–124.

Kenneth W. Church and William A. Gale. 1995. Poisson mixtures. *Nat. Lang. Eng.* 1, 2 (1995), 163–190.

Kenneth Ward Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. *Comput. Ling.* 16, 1 (Mar. 1990), 22–29.

Rudi L. Cilibrasi and Paul M. B. Vitányi. 2005. Clustering by compression. *IEEE Trans. Inf. Theory* 51, 4 (Apr. 2005), 1523–1545.

Rudi L. Cilibrasi and Paul M. B. Vitányi. 2007. The google similarity distance. *IEEE Trans. Knowl. Data Eng.* 19, 3 (Mar. 2007), 370–383.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Mach. Learn.* 20, 3 (Sept. 1995), 273–297.

Joaquim Ferreira da Silva and Jos'e Gabriel Pereira Lopes. 1999. A local maxima method and a fair dispersion normalization for extracting multi-word units from corpora. In *Proceedings of Meeting on Mathematics of Language (MOL)*. 369–381.

Erik D. Demaine, Alejandro López-Ortiz, and J. Ian Munro. 2000. Adaptive set intersections, unions, and differences. In *Proceedings of ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 743–752.

Gaël Dias. 2000. Mining textual associations in text corpora. In *Proceedings of ACM SIGKDD Workshop on Text Mining*. 20–23.

Richard Durstenfeld. 1964. Algorithm 235: Random permutation. *Commun. ACM* 7, 7 (July 1964), 420.

Benjamin C. M. Fung, Ke Wang, and Martin Ester. 2003. Hierarchical document clustering using frequent itemsets. In *Proceedings of SIAM International Conference on Data Mining (SDM)*. 59–70.

Travis Gagie, Gonzalo Navarro, and Simon J. Puglisi. 2012. New algorithms on wavelet trees and applications to information retrieval. *Theor. Comput. Sci.* 426–427 (Apr. 2012), 25–41.

Warren R. Greiff. 1998. A theory of term weighting based on exploratory data analysis. In *Proceedings of International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 11–19.

Roberto Grossi, Ankur Gupta, and Jeffrey Scott Vitter. 2003. High-order entropy-compressed text indexes. In *Proceedings of ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 841–850.

Matthias Hagen, Martin Potthast, Benno Stein, and Christof Bräutigam. 2011. Query segmentation revisited. In *Proceedings of International World Wide Web Conference (WWW)*. 97–106.

Frank A. Haight. 1967. *Handbook of the Poisson Distribution*. Wiley, New York.

Stephen P. Harter. 1975. A probabilistic approach to automatic keyword indexing. Part I. On the distribution of specialty words in a technical literature. *J. Am. Soc. Inf. Sci.* 26, 4 (July/Aug. 1975), 197–206.

Kazi Saidul Hasan and Vincent Ng. 2010. Conundrums in unsupervised keyphrase extraction: Making sense of the state-of-the-art. In *Proceedings of International Conference on Computational Linguistics (COLING)*. 365–373.

Djoerd Hiemstra. 2000. A probabilistic justification for using TF×IDF term weighting in information retrieval. *Int. J. Dig. Libr.* 3, 2 (Aug. 2000), 131–139.

Karen Spärck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *J. Doc.* 28 (1972), 11–21.

Andrey Kolmogorov. 1963. On tables of random numbers. *Sankhyā Ser. A* 25 (1963), 369–376.

Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. 2004. Applying conditional random fields to Japanese morphological analysis. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 230–237.

Man Lan, Chew Lim Tan, Jian Su, and Yue Lu. 2009. Supervised and traditional term weighting methods for automatic text categorization. *IEEE Trans. Pattern Anal. Mach. Intell.* 31, 4 (Apr. 2009), 721–735.

Rolf Landauer. 1961. Irreversibility and heat generation in the computing process. *IBM J. Res. Dev.* 5, 3 (July 1961), 183–191.

Ken Lang. 1995. NewsWeeder: Learning to filter netnews. In *Proceedings of International Conference on Machine Learning (ICML)*. 331–339.

Ming Li and Paul M. B. Vitányi. 2009. *An Introduction to Kolmogorov Complexity and Its Applications* (3rd ed.). Springer, Berlin.

Zhiyuan Liu, Peng Li, Yabin Zheng, and Maosong Sun. 2009. Clustering to find exemplar terms for keyphrase extraction. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 257–266.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press, Cambridge.

Donald Metzler. 2008. Generalized inverse document frequency. In *Proceedings of ACM Conference on Information and Knowledge Management (CIKM)*. 399–408.

Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing order into texts. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 404–411.

Nikita Mishra, Rishiraj Saha Roy, Niloy Ganguly, Srivatsan Laxman, and Monojit Choudhury. 2011. Unsupervised query segmentation using only query logs. In *Proceedings of International World Wide Web Conference (WWW)*. 91–92.

Kazuyuki Narisawa, Shunsuke Inenaga, Hideo Bannai, and Masayuki Takeda. 2007. Efficient computation of substring equivalence classes with suffix arrays. In *Proceedings of Symposium on Combinatorial Pattern Matching (CPM)*. 340–351.

Daisuke Okanohara and Jun'ichi Tsujii. 2009. Text categorization with all substring features. In *Proceedings of SIAM International Conference on Data Mining (SDM)*. 838–846.

Constantin Orăsan, Viktor Pekar, and Laura Hasler. 2004. A comparison of summarisation methods based on term specificity estimation. In *Proceedings of International Conference on Language Resources and Evaluation (LREC)*. 1037–1041.

Kishore Papineni. 2001. Why inverse document frequency? In *Proceedings of Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL)*. 1–8.

Pavel Pecina. 2005. An extensive empirical study of collocation extraction methods. In *Proceedings of ACL Student Research Workshop*. 13–18.

Xuan-Hieu Phan, Le-Minh Nguyen, and Susumu Horiguchi. 2008. Learning to classify short and sparse text & web with hidden topics from large-scale data collections. In *Proceedings of International World Wide Web Conference (WWW)*. 91–100.

Joel W. Reed, Yu Jiao, Thomas E. Potok, Brian A. Klump, Mark T. Elmore, and Ali R. Hurson. 2006. TF-ICF: A new term weighting scheme for clustering dynamic data streams. In *Proceedings of International Conference on Machine Learning and Applications (ICMLA)*. 258–263.

Jason D. M. Rennie and Tommi Jaakkola. 2005. Using term informativeness for named entity detection. In *Proceedings of International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 353–360.

Stephen Robertson. 2004. Understanding inverse document frequency: On theoretical arguments for IDF. *J. Doc.* 60, 5 (2004), 503–520.

Stephen Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. 1994. Okapi at TREC-3. In *Proceedings of Text Retrieval Conference (TREC)*. 109–126.

Thomas Roelleke and Jun Wang. 2008. TF-IDF uncovered: A study of theories and probabilities. In *Proceedings of International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 435–442.

Francois Rousseau and Michalis Vazirgiannis. 2013. Graph-of-word and TW-IDF: New approach to Ad Hoc IR. In *Proceedings of ACM Conference on Information and Knowledge Management (CIKM)*. 59–68.

Rishiraj Saha Roy, Niloy Ganguly, Monojit Choudhury, and Srivatsan Laxman. 2012. An IR-based evaluation framework for web search query segmentation. In *Proceedings of International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 881–890.

Rishiraj Saha Roy, Yogarshi Vyas, Niloy Ganguly, and Monojit Choudhury. 2014. Improving unsupervised query segmentation using parts-of-speech sequence information. In *Proceedings of International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 935–938.

Gerard Salton and Michael J. McGill. 1983. *Introduction to Modern Information Retrieval*. McGraw-Hill, New York.

Gerard Salton, Anita Wong, and Chung-Shu Yang. 1975. A vector space model for automatic indexing. *Commun. ACM* 18, 11 (Nov. 1975), 613–620.

Patrick Schone and Daniel Jurafsky. 2001. Is knowledge-free induction of multiword unit dictionary headwords a solved problem? In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 100–108.

Masumi Shirakawa, Takahiro Hara, and Shojiro Nishio. 2015. N-gram IDF: A global term weighting scheme based on information distance. In *Proceedings of International World Wide Web Conference (WWW)*. 960–970.

Josef Sivic and Andrew Zisserman. 2003. Video google: A text retrieval approach to object matching in videos. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*. 1470–1477.

Mika Timonen. 2013. *Term Weighting in Short Documents for Document Categorization, Keyword Extraction and Query Expansion*. Ph.D. Dissertation. University of Helsinki.

Takashi Tomokiyo and Matthew Hurst. 2003. A language model approach to keyphrase extraction. In *Proceedings of ACL Workshop on Multiword Expressions: Analysis, Acquisition and Treatment*. 33–40.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*. 252–259.

Peter D. Turney. 2003. Coherent keyphrase extraction via web mining. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*. 434–439.

Gerald van Belle, Lloyd D. Fisher, Patrick J. Heagerty, and Thomas Lumley. 2004. *Biostatistics: A Methodology For the Health Sciences* (2nd ed.). Wiley, New York.

Xiaojun Wan and Jianguo Xiao. 2008. Single document keyphrase extraction using neighborhood knowledge. In *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*. 855–860.

Ian H. Witten, Gordon W. Paynter, Eibe Frank, Carl Gutwin, and Craig G. Nevill-Manning. 1999. KEA: Practical automatic keyphrase extraction. In *Proceedings of ACM Conference on Digital Libraries (DL)*. 254–255.

Ho Chung Wu, Robert Wing Pong Luk, Kam Fai Wong, and Kui Lam Kwok. 2008. Interpreting TF-IDF term weights as making relevance decisions. *ACM Trans. Inf. Syst.* 26, 3 (June 2008), 13:1–13:37.

Ting-Fan Wu, Chih-Jen Lin, and Ruby C. Weng. 2004. Probability estimates for multi-class classification by pairwise coupling. *J. Mach. Learn. Res.* 5 (2004), 975–1005.

Mikio Yamamoto and Kenneth W. Church. 2001. Using suffix arrays to compute term frequency and document frequency for all substrings in a corpus. *Comput. Ling.* 27, 1 (Mar. 2001), 1–30.

Wen Zhang, Taketoshi Yoshida, Xijin Tang, and Tu-Bao Ho. 2009. Improving effectiveness of mutual information for substantival multiword expression extraction. *Expert Syst. Appl.* 36, 8 (Oct. 2009), 10919–10930.

Received January 2016; revised November 2016; accepted December 2016