Expert Systems with Applications 40 (2013) 1467–1476

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Detecting near-duplicate documents using sentence-level features and supervised learning

Yung-Shen Lin, Ting-Yi Liao, Shie-Jue Lee *

Department of Electrical Engineering, National Sun Yat-Sen University, Kaohsiung 804, Taiwan

ARTICLE INFO

Keywords:
Near-duplicate
Feature selection
Similarity function
Training data
Support vector machine
Discriminant function

ABSTRACT

We present a novel method for detecting near-duplicates from a large collection of documents. Three major parts are involved in our method, feature selection, similarity measure, and discriminant derivation. To find near-duplicates to an input document, each sentence of the input document is fetched and preprocessed, the weight of each term is calculated, and the heavily weighted terms are selected to be the feature of the sentence. As a result, the input document is turned into a set of such features. A similarity measure is then applied and the similarity degree between the input document and each document in the given collection is computed. A support vector machine (SVM) is adopted to learn a discriminant function from a training pattern set, which is then employed to determine whether a document is a near-duplicate to the input document based on the similarity degree between them. The sentence-level features we adopt can better reveal the characteristics of a document. Besides, learning the discriminant function by SVM can avoid trial-and-error efforts required in conventional methods. Experimental results show that our method is effective in near-duplicate document detection.

1. Introduction

As the World Wide Web is increasingly popular, digital documents are easily generated and put on the internet. By using a search engine, one can collect a large set of documents in a very short time (Chowdhury, Frieder, Grossman, & McCabe, 2002; Henzinger, 2006). Through the delete, copy, and paste commands provided by an editor or other tools (de Carvalho, Laender, Goncalves, & da Silva, 2012; Valls & Rosso, 2011), similar documents are likely to appear in various web communities (Conrad, Guo, & Schriber, 2003; Fetterly, Manasse, & Najork, 2003; Manku, Jain, & Sarma, 2007; Narayana, Premchand, & Govardhan, 2009; Pereira, Baeza-Yates, & Ziviani, 2006; Yang & Callan, 2005), e.g., blogs and forums. Such similar documents not only increase the volume of information one may have to go through but also require more storage and bandwidth for communication. To reduce the data volume and increase the search efficiency, detecting similar documents has become an important issue in the field of information retrieval (Pereira et al., 2006).

Similar documents can be divided into two categories, duplicates and near-duplicates. Two documents are duplicates if they are totally identical (Broder, 2000). Two documents are near-duplicates if one document is a modification of the other document. The modification can be insertion, deletion, or replacement of parts of the text. Due to the provision of editing facilities, near-duplicate documents are prevailing on almost all kinds of social media (Enron email dataset, 2012; Yang & Callan, 2006). Duplicate documents can be easily detected. However, detecting near-duplicates is much harder (Sood & Loguinov, 2011; Jiang & Sun, 2011). In this paper, we focus on how to detect near-duplicate documents efficiently and effectively.

To detect near-duplicate documents, one can adopt the bag-of-words model (Bag of words, 2012) for document representation. Let $D = \{d_1, d_2, \ldots, d_n\}$ be a set of n documents, in which $d_1, d_2, \ldots, d_n$ are individual documents. Each document d_i, $1 \leqslant i \leqslant n$, is represented by a feature set $f_i = \{f_{i,1}, f_{i,2}, \ldots, f_{i,m}\}$ where m is the number of features selected for D. A feature can be anything associated with documents (Arasu, Ganti, & Kaushik, 2006; Fagin, Kumar, & Sivakumar, 2003; Gong, Huang, Cheng, & Bai, 2008; Huffman et al., 2007; Li, Wang, & Yang, 2007; Qiu & Zeng, 2010; Theobald, Siddharth, & Paepcke, 2008; Wang & Chang, 2009; Xiao, Wang, Lin, & Shang, 2009, 2011; Yang & Callan, 2006). Usually, terms appearing in documents are taken to be the basis of features (Goyal, Behera, & McGinnity, 2012; Han, Finin, McNamee, Joshi, & Yesha, 2012; Kim & Lee, 2012; Luo, Lin, Wang, & Zhou, 2007). To investigate how one document is similar to another document, one can calculate the similarity degree between the two sets of features corresponding to these two documents (Bayardo, Ma, & Srikant, 2007; Huang, Wang, & Li, 2008; Zhao, Wang, Liu, & Ye, 2011). The higher the degree is, the more the documents are similar to each other. Conventionally, a manually designated

* Corresponding author.
E-mail address: leesj@mail.ee.nsysu.edu.tw (S.-J. Lee).

0957-4174/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.eswa.2012.08.045

Table 1
An example list of stop words.

a, about, above, after, again, against, all, am, an, and, any, are, aren't, as, at, be, because, been, before, being, below, between, both, but, by, can't, cannot, could, couldn't, did, didn't, do, does, doesn't, doing, don't, down, during, each, few, for, from, further, had, hadn't, has, hasn't, have, haven't, having, he, he'd, he'll, he's, her, here, here's, hers, herself, him, himself, his, how, how's, I, i'd, i'll, i'm, i've, if, in, into, is, isn't, it, it's, its, itself, let's, me, more, most, mustn't, my, myself, no, nor, not, of, off, on, once, only, or, other, ought, our, ours

threshold is provided by the user in advance. If the similarity degree is equal to or higher than the threshold, the two documents are near-duplicates. Otherwise, they are not. However, different choices on features may require different settings of the threshold. Besides, the determination of a satisfactory threshold is also a problem. Usually, trial-and-error cannot be avoided. Setting a good threshold manually is neither an easy task nor an effective way for near-duplicate document detection.

We develop a novel method for detecting near-duplicates from a large collection of documents. Our method consists of three major components, feature selection, similarity measure, and discriminant derivation. To find near-duplicates to an input document, we first do preprocessing, e.g., removing stop words and punctuation marks, on the input document. Then for each sentence, the weight of each term is calculated, and the heavily weighted terms are selected to be the feature of the sentence. As a result, the input document is turned into a set of features. Then the similarity degree between the input document and each document in the given collection is computed. Finally, we use a support vector machine to learn a classifier from a training pattern set (Arnosti & Kalita, 2011; Brin, Davis, & Garcia-Molina, 1995; Hajishirzi, Yih, & Kolcz, 2010; Martins, 2011). A discriminant function is derived, which is then used to determine whether a document is a near-duplicate to the input document based on the similarity degree between them. Our method has several advantages. The sentence-level features we adopt can better reveal the characteristics of a document, and learning the discriminant function by SVM can avoid trial-and-error efforts required in conventional methods. Experimental results show that our method is effective in near-duplicate document detection.

The rest of this paper is organized as follows. Section 2 presents a brief description about related work. Section 3 details our proposed near-duplicate document detection method. The way to create a feature set for a given document and the adoption of SVM in learning a discriminant function are described. Experimental results are presented in Section 4. Comments on a frequency-based representation are discussed in Section 5. Finally, a conclusion is given in Section 6.

2. Related work

For a document, Shingles (Manning, Raghavan, & Schutze, 2008) divided it into a series of strings. Each string is k words long, called a k-gram. The list of such k-grams is taken to be the feature set of this document, This method may result in a large feature set. For example, if a document consists of L words, the feature set of the document contains $L-k+1$ elements. Some improvements to Shingles have been proposed. Li et al. (2007) took discontinuous k-grams by skipping the words in between. The strings between two pause symbols are treated as features. The SpotSigs proposed by Theobald et al. (2008) used stop words (Common Stopword set, 2012) instead. A feature is taken to be a string starting with a stop word. For example, {*the super computer*} and {*a good movie*} are elements of the feature set. However, the stop word list adopted is a key factor to the feature set obtained. Different stop word lists lead to different feature sets for a given document. A popular stop word list used in many applications is shown in Table 1. SpotSigs (Theobald et al., 2008) adopts some rules to cut down the size of a feature set, e.g., preferring more frequently used stop words. Other methods based on sentences were proposed (Wang & Chang, 2009). With these methods, each individual sentence of a document is divided into a series of k-grams. The union of the k-grams of all the sentences is taken as the feature set of the document. However, these methods result in large feature sets for document representation.

A similarity function is used to calculate the similarity degree of any two documents. Let $f_1=\{f_{1,1},f_{1,2},\ldots,f_{1,m}\}$ and $f_2=\{f_{2,1},f_{2,2},\ldots,f_{2,m}\}$ be the feature sets of documents d_1 and d_2, respectively. Some popular similarity functions are listed below.

- Jaccard function:

$$sim(d_1,d_2)\equiv J(d_1,d_2)=\frac{f_1\cap f_2}{f_1\cup f_2} \tag{1}$$

  where $\cap$ stands for the AND operation and $\cup$ for the OR operation in the set theory.
- Cosine function:

$$sim(d_1,d_2)\equiv C(d_1,d_2)=\frac{f_1\cdot f_2}{\|f_1\|\cdot\|f_2\|} \tag{2}$$

  where $f_1\cdot f_2$ is defined to be $f_1\cdot f_2=f_{1,1}f_{2,1}+f_{1,2}f_{2,2}+\ldots+f_{1,m}f_{2,m}$, and $\|f_i\|=\sqrt{f_i\cdot f_i}$ for $i=1,2$.
- Euclidean distance:

$$sim(d_1,d_2)\equiv Ec(d_1,d_2)=\sqrt{(f_1-f_2)\cdot(f_1-f_2)} \tag{3}$$

  where $f_1-f_2=\{f_{1,1}-f_{2,1},f_{1,2}-f_{2,2},\ldots,f_{1,m}-f_{2,m}\}$.
- Extended Jaccard function:

$$sim(d_1,d_2)\equiv EJ(d_1,d_2)=\frac{f_1\cdot f_2}{f_1\cdot f_1+f_2\cdot f_2-f_1\cdot f_2}. \tag{4}$$

- Dice function:

$$sim(d_1,d_2)\equiv D(d_1,d_2)=\frac{2f_1\cdot f_2}{f_1\cdot f_1+f_2\cdot f_2}. \tag{5}$$

Note that $EJ(d_1,d_2)$ is an extended version of $J(d_1,d_2)$ and $D(d_1,d_2)$ is a simplified version of $EJ(d_1,d_2)$.

Conventionally, a manually pre-designated threshold provided by the user is required to determine if two documents are near-duplicates. Supervised learning techniques, in particular support vector machines (SVM) (Martins, 2011), can be applied to determine optimally whether two documents are near-duplicates automatically. Given a training data set with instances belonging to one of two classes, near-duplicate and non-near-duplicate, SVM learns how to separate the instances of one class from the instances of the other class. As matter of fact, an optimal hyperplane can be derived which not only separates the instances on the right side of the hyperplane but also maximizes the margin from the hyperplane to the instances closest to it on either side. If the problem is not linearly separable, one can map the original space to a new space by using nonlinear basis functions. It is generally the case that this new space has many more dimensions than the original space, and, in the new space, the optimal hyperplane can be found.