Electronic Preprint for Journal of Information Processing Vol.28

Regular Paper

Detection of Mergeable Wikipedia Articles Utilizing Multiple Similarity Measures

Renzhi Wang[1,a)] Mizuho Iwaihara[1]

Received: June 9, 2019, Accepted: October 2, 2019

Abstract: Wikipedia is the largest online encyclopedia, in which articles are edited by different volunteers with different thoughts and styles. Sometimes two or more articles' titles are different but the themes of these articles are exactly the same or strongly similar. Administrators and editors are supposed to detect such article pairs and determine whether they should be merged together. We call an article pair is mergeable if it is discussed for possible merge, and a merged article pair is such that the pair is actually merged. In this paper, we propose a method to automatically determine whether an article pair is mergeable or merged. According to Wikipedia Guidelines for article merge, in the duplicate case, the article pairs are covering exactly the same contents. In the overlap case, the article pairs are covering related subjects that have a significant overlap. The content of an overlapped part is similar but the words in the pair can be extensively different, so methods that exploit semantic relatedness are necessary. We consider various textual similarities and semantic relatedness. For integrating word embeddings on the target dataset and the global large corpus, we propose linear and non-linear combinations of multiple embedding results and rebuilding word vectors for evaluating semantic relatedness. We clarify the differences between our method and previous researches for combining multiple word embeddings. We also deal with overlap cases by computing Jaccard similarity between article pairs. We combine Jaccard similarity, common-link article count and word embedding-based relatedness together, to predict whether the article pair should be merged. We explore the relationship between segment-level (paragraph-level) similarity and mergeable/merged article pairs, then propose Multimodal Similarity-Based Merge Prediction (MSBMP) which combines the proposed new features by Random Forest, to predict mergeable/merged article pairs. Our evaluations are performed on real mergeable and merged article pairs. Remarkable superiorities of MSBMP are shown, with apparent improvement from baselines of WikiSearch, TFIDF and word embeddings.

Keywords: word embedding, mergeable article, Wikipedia, text mining

1. Introduction

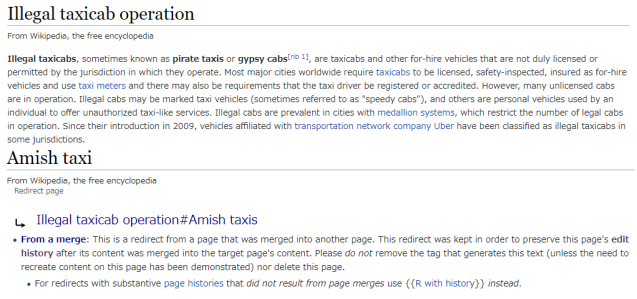
Wikipedia articles are edited by various volunteers from all over the world. Each article in Wikipedia identifies a clear concept. Due to diverse culture and cognition backgrounds, one concept may be written in various styles by different editors in different articles. In Wikipedia, detection of near duplicate texts is necessary for copyright enforcement and for helping version management. Administrators and editors need to find and merge duplicate articles to avoid confusing readers and maintain integrity. As stated in the Wikipedia guidelines for merge, there are four reasons to merge articles [26]: 1) Duplicate: There are two or more pages on exactly the same subject, with the same scope. 2) Overlap: There are two or more pages on related subjects that have a large overlap. 3) Text: If a page is very short and is unlikely to be expanded within a reasonable amount of time, it often makes sense to merge it with a page on a broader topic. 4) Context: If a short article requires the background material or context from a broader article in order for readers to understand it.

One example of article merge is that the articles "Chickens and dumplings" and "Bott boi," shown in **Fig. 1**, are suggested to be merged together on August 1st, 2018. One of the reasons is that these two dishes use the same cooking materials, and their recipes are similar. The people who disagree on merging these two articles comment that these two dishes taste different. We call an article pair to which merge is proposed as a *mergeable* article pair. Another case is the articles "Illegal taxicab operation" and "Amish taxi," shown in **Fig. 2**. They were actually merged together after discussion, and "Amish taxi" is already redirected to "Illegal taxicab operation." We call these cases as *merged* article pairs. In this paper, we focus on two tasks such that, the first is to detect mergeable article pairs from candidate similar pairs, and the second is to detect article pairs that should be actually merged.

Fig. 1 Mergeable Wikipedia Articles.

Currently the Wikipedia article merge task is done by human editors through discussion. Editors firstly create a talk page to

[1] Waseda University, Kitakyushu, Fukuoka 808–0135, Japan
[a)] ouninnyuki.ips@asagi.waseda.jp


Illegal taxicab operation

From Wikipedia, the free encyclopedia

Illegal taxicabs, sometimes known as **pirate taxis** or **gypsy cabs**[no 1], are taxicabs and other for-hire vehicles that are not duly licensed or permitted by the jurisdiction in which they operate. Most major cities worldwide require taxicabs to be licensed, safety-inspected, insured as for-hire vehicles and use taxi meters and there may also be requirements that the taxi driver be registered or accredited. However, many unlicensed cabs are in operation. Illegal cabs may be marked taxi vehicles (sometimes referred to as "speedy cabs"), and others are personal vehicles used by an individual to offer unauthorized taxi-like services. Illegal cabs are prevalent in cities with medallion systems, which restrict the number of legal cabs in operation. Since their introduction in 2009, vehicles affiliated with transportation network company Uber have been classified as illegal taxicabs in some jurisdictions.

Amish taxi

From Wikipedia, the free encyclopedia
Redirect page

↳ Illegal taxicab operation#Amish taxis

- **From a merge**: This is a redirect from a page that was merged into another page. This redirect was kept in order to preserve this page's **edit history** after its content was merged into the target page's content. Please *do not* remove the tag that generates this text (unless the need to recreate content on this page has been demonstrated) nor delete this page.
  - For redirects with substantive page histories that *did not result from page merges* use {{R with history}} *instead*.

Fig. 2 Merged Wikipedia Articles.

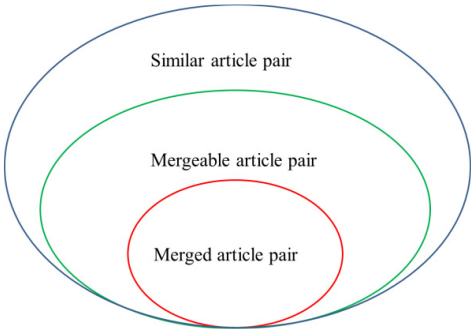


Fig. 3 Containment relationship between mergeable article pairs and merged article pairs.

discuss whether an article pair should be merged. If most of the editors agree to merge the article pair, then the pair will be merged. So there exist two stages for merging articles. In the first stage two articles are proposed to be merged but still under discussion, and in the second stage the two articles are actually merged. Therefore, there exist containment relationships between similar, mergeable, and merged article pairs, as depicted in **Fig. 3**. In real situations, the number of similar article pairs is much higher than the number of mergeable article pairs. Most of similar article pairs are not considered for possible merge. The number of pairs actually merged after editors' agreement is further reduced. Here we show a real example of this situation. When we executed query "3D printing" on WikiSearch, 27,554 results were returned. These results are regarded as similar to article "3D printing." But in the talk page of this article, only three articles "3D printer," "additive manufacturing," and "rapid prototyping" were mergeable, namely they were discussed to be merged with "3D printing." These three articles were mergeable articles at the query time, but after discussion, only "3D printer" and "additive manufacturing" were merged into "3D printing," becoming merged article pairs. Article "rapid prototyping" was rejected to be merged with "3D printing."

Finding articles to be merged is similar to the near-duplicate text detection problem. For a large collection of documents, evaluating a query over every document in the collection is too costly, so conventional approaches focus on how to initially select fewer candidates. After obtaining a small set of candidates, two documents are compared closely by term co-occurrence similarity. Previous researches [2], [5], [6], [7], [16] focus on clustering similar documents over a large corpus. For plagiarism detection, word co-occurrence-based features are quite important in detecting duplicate texts. On the other hand, our goal is further detecting article pairs which are not just heavily duplicated or having a significant overlap, but also topic containment and context containment, as stated in the Wikipedia guidelines for merge. Our candidate merge pairs can be simply retrieved by a conventional retrieval like WikiSearch, so we put more emphasis on determining whether candidate similar pairs should be merged. In Wikipedia, various editors often use varieties of expressions in writing an identical article, although their intensions are basically the same. Term co-occurrence similarity is not sufficient for the case of diverse wordings with the same intension.

In this paper, we discuss integration of textual, semantic and link-structural similarities, to determine whether a given article pair is mergeable or merged. For semantic similarity, we adopt the well-known word embedding method, word2vec [18], [19]. The difficulty of our task is that, the known pre-trained embedding result is based on a very large corpus. Compared with such a corpus, our target dataset is just a portion of Wikipedia, so the distribution of word occurrence can be distinctively apart from the general distribution. So directly using pre-trained embedding causes undesirable results, such as words having a specific meaning in our target dataset. Directly using our target dataset to train a specific embedding result is also undesirable, because compared with large corpora, our dataset is too small to train a good embedding result. To solve this problem, we propose linear combination and non-linear combination of multiple pre-trained embeddings to train customized embedding results. In the linear combination, we utilize transfer matrices, motivated by translation matrix in Ref. [20], to combine multiple pre-trained embedding results. In non-linear combination, we utilize a neural network to combine multiple pre-trained embedding results. The main difference between our methods and the previous approaches is that we define a new loss function and retrain the embedding results over our target dataset.

Our approach is motivated by transductive transfer learning [12]. The concept of transductive transfer learning is that the source domain (*Ds*) and source domain task (*Ts*) are given, and the target domain (*Dt*) and target domain task (*Tt*) are the goal. Here *Ts* is equal to *Tt*, but *Ds* is not equal to *Dt*. The transductive transfer learning methods utilize the knowledge of the source domain and source domain task, to improve the prediction function for the target domain and target domain task. Usually, the source domain task is over a large labeled dataset, while the target domain task has only a limited labeled dataset. In our case, the pre-trained embedding results are the source domain and source task, and a given set of candidate article pairs for merge is the target domain. We propose a new loss function to improve the embedding results for our mergeable article dataset.

To be accordant with the criteria for merge of articles in Wikipedia, we utilize both Jaccard distance and semantic similarity to evaluate surface-level and semantic-level similarities. Furthermore, since each article has a certain length and only a part of one article may be overlapping with another article, we introduce features on similarity distributions over article segments, where *segments* mean any logical unit of articles, and in this paper we adopt paragraphs as segments. We call the new method combining all the proposed features by Random Forest as *Multimodal Similarity-Based Merge Prediction* (*MSBMP*).

Our experiments on both real Wikipedia mergeable articles and merged articles show that our method MSBMP predicts better than subsets of proposed features, and baselines such as WikiSearch, TFIDF, and global word embeddings trained over large corpora, in terms of AVGRecall@K, mean reciprocal rank (MRR), and area under the ROC curve (AUC).

The rest of the paper is organized as follows: Section 2 covers related work. Section 3 shows our proposing method. In Section 4, we describe our datasets in detail, and explain our experimental process. In Section 5, evaluation results on the proposed feature combination methods are shown. Section 6 studies the impact of overlap size and article length on our task. Section 7 concludes this paper.

2. Related Work

For near duplicate text detection, a variety of signature generation methods, encompassing scalability, have been proposed. Previous researches [5], [6], [7], [16] separately proposed shingling-based, windowing-based, simhash-based algorithms to detect near duplicate texts. But these methods only exploit co-occurring terms, where semantic relatedness is not considered. These methods cannot handle texts that use a large number of different terms but expressing the same topic.

Weissman et al. [23] propose a minhash-based method over MapReduce to detect near-duplicated texts in Wikipedia. Their approach detects near-duplicated texts in the sentence level. On the other hand, our task of detecting merged articles needs to detect duplicates at the article level, where overlaps are often limited to narrow segments.

Recent researchers consider incorporating semantic information into document signatures. Alonso et al. [2] consider TF-IDF weighting in their signature algorithm, to reflect certain semantic information.

Schofield et al. [17] show learning on how repeated texts affect semantic models. They trained a Latent Dirichlet Allocation model and Latent Semantic Analysis model over different levels of repeated texts. They discussed observing perplexity during training progress and recommending a suitable model for different levels of repeated texts in a corpus. In this paper, we also discuss the effect of varying overlaps, and discuss its suitable treatment.

For plagiarism detection, word co-occurrence-based features are very important in detecting duplicate texts. Pertile et al. [15] utilized the Jaccard distance of article pairs and co-occurrences in citations to detect plagiarism papers. But only word co-occurrence-based features cannot detect semantic information well.

Word embeddings are becoming an effective way to represent words by relatively low-dimensional vectors, where semantic relatedness is easily measured by the cosine similarity of two word vectors. Ferrero et al. [8] introduce a syntax weighting in distributed representations of sentences, and prove its usefulness for textual similarity detection. But in their approach, they only utilized one large dataset to build word vectors.

To integrate multiple embedding results, Mikolov et al. [20] use translation matrices to convert embeddings in English into embeddings in Spanish. Peng et al. [14] redefine nodes of hierarchical structure (in the word2vec model it is Huffman tree), where each node shares the data from global corpus and target corpus. Their model retrains the hierarchical tree and produces final word vectors. But a disadvantage is that their model needs to be trained on both global corpus and target corpus. Garten et al. [9] test the sum and concatenation of the two vectors for a given word, for combining multiple embedding results. This method shows an improvement but it is too simple and their obtained results do not learn from the target corpus. Their model loses the distribution information of the target corpus. Yin et al. [24] compare five ensemble methods to combine multiple embedding results. They employ concatenation-based, singular value decomposition-based methods, and 1 to N, 1 to N+ strategies to build ensemble models. They also propose mutual learning to extend these four models.

In our method, we intend to combine pre-trained embedding results to generate a new embedding result over the target corpus. So the essential difference between our method and Garten's and Yin's models is such that what their method trains is an ensemble model rather than an embedding model, while our method trains an embedding model over the target corpus. Peng's model is also training a new embedding, but their method needs a global dataset, and their method must build a new large hierarchical tree. Compared with this model, our proposing method only needs a target corpus and a pre-trained embedding result, not requiring a large global corpus.

3. Proposed Method

3.1 Overview of Proposed Method

In this paper, we mainly investigate solutions for the following two target tasks:

Task 1 (Mergeable articles): Given a target Wikipedia article, find articles which are mergeable with the target article. If there is no mergeable article in the candidate set, a null result (no mergeable articles) is returned.

Task 2 (Merged articles): Given a target Wikipedia article, find articles that should be merged with the target article.

Since one article can be quite long, we divide one article into *segments*, where a segment is any logical component of articles, and in this paper we choose paragraphs as segments. We measure similarities between two articles by both the *article level* (whole article) and *segment level* (every pair of their segments).

The flow diagram of our method is shown in **Fig. 4**.

For Tasks 1 and 2, we consider returning a confidence score of a given article pair being mergeable or merged. We also investigate an optimum threshold on the confidence score, for predicting whether or not article pairs are mergeable/merged.

To solve these two problems, we propose combination of a variety of features, where one feature is based on multiple embeddings for evaluating the semantic relatedness of article pairs. Also, another feature is based on Jaccard distance over both the article level and the segment level to evaluate the word overlap between two articles. Segment similarity distribution characterizes distribution of similarity values between all the segment pairs within two articles. To incorporate relatedness by the link structure of Wikipedia, we consider common link-to article counts as

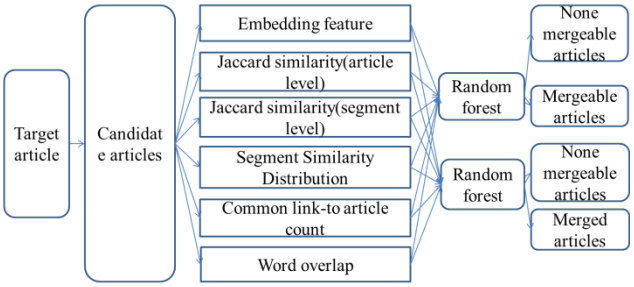


Fig. 4 Flow diagram of proposed model.

a feature. Then we utilize Random Forest [4] to generate a classifier for determining mergeable article pairs in candidate pairs, as well as for producing a confidence score on each classification result for ranking.

For Task 2 on merged article pairs, we adopt the same proposed features as Task 1, but a Random Forest classifier is trained over a different set of article pairs which are labeled with either merged or not.

In real situations, mergeable article pairs can be eventually merged after a period of discussion. But in our evaluation, we can collect a reference dataset from edit histories only at a collection time point, where future merges are unknown. Therefore, in our evaluation we assume that merged and mergeable articles are limited to those recorded in the edit histories.

From the Wikipedia guideline, we can expect that merged articles share similar topics, common words, and/or common links. Thus we design our algorithm based on these three properties. We measure semantic similarity by using embedding-based methods. For segment-level similarity, we introduce segment-level Jaccard similarity to evaluate common words. We also utilize common link-to articles, which are articles co-cited by article pairs, for measuring relatedness by the link structure. Next, we describe our method in detail.

3.2 Linear Combination of Word Embeddings

Word embeddings are compact vector representation of words, becoming the most effective way to measure semantic similarity between words [18], [19]. Word2vec assumes two language models, Continuous Bag of Words (CBOW) [3] and Skip-gram [18]. The CBOW model assumes that context words' vectors should predict the target word, while the Skip-gram model assumes that the target word should predict the context words. Based on these assumptions, objective functions are defined as products of all target words' predicted probabilities, as follows:

$$Objective = \sum_{w \in C} log\, P(w \mid context(w)) \quad \text{– CBOW}$$

$$Objective = \sum_{w \in C} log\, P(context(w) \mid w) \quad \text{– Skip-gram}$$

To perform scalable training over large datasets, word2vec utilizes Huffman trees for maximizing the objective function. After training, we can obtain distributed word vector representations, which will be used to compute similarities between article pairs.

Our goal needs to deal with a small training dataset of mergeable articles. To combine pre-trained embedding results, we utilize transfer matrices to integrate multiple embedding results. We also define the sum of all the embedding results multiplied by

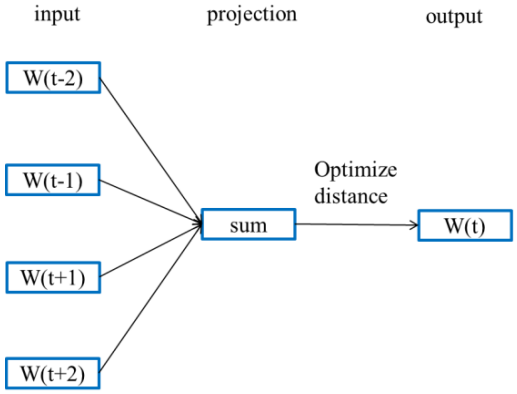


Fig. 5 Proposed object function.

transfer matrices as the final embedding result, defined as:

$$E_f = \sum_{i=1}^{n} E_i M_i \tag{1}$$

Here, E_i, $i = 1, \ldots, n$, is a pre-trained embedding result and M_i, $i = 1, \ldots, n$, is a transfer matrix, and E_f is the final embedding result.

To fit for the target dataset, we define a new loss function. As the original word2vec model assumes, we also suppose that context words can predict a target word. In the embedding space, this assumption can be regarded as the average of context words should be the closest to the target word and the average of context words should be far away from the other words.

Based on this assumption, we define the objective function as follows:

$$Objective = \sum_{w \in C} Dis(context(w), w) \tag{2}$$

Here function *Dis* is the distance between the sum of context word vectors and target word vector and C is the corpus. Here the distance function can be any reasonable distance such as Manhattan distance, Euclidean distance, cosine similarity, etc. In our case, we use Euclidean distance for the *Dis* function. The difference between our objective function and CBOW is that we use Euclidean distance as our *Dis* function. There are two advantages of using Euclidean distance. One is that for small datasets, we do not need to build a complex hierarchal softmax layer (in the word2vec model that is a Huffman tree) to compute the probabilities. The other is that to measure the similarity of vectors, compared with cosine similarity which just measures the angle between vectors, Euclidean distance measure the absolute distance. In Mikolov's research [20], the authors also utilized the Euclidean distance as the loss function. So we directly compute the Euclidean distance between a context word vector and target word vector.

Below we explain how we compute the transfer matrices. As our objective function uses the Euclidean distance, we can rewrite the function as:

$$loss = \sum_{w_j \in Vocabulary} \|v_{c_j} - v_{w_j}\|_2 \tag{3}$$

Here w_j is the j^{th} word in the vocabulary from the target corpus, and v_{w_j} is the vector of w_j. Similarly, v_{c_j} is the j^{th} context vector, where context is the other words in w_j's window.

We represent v_{c_j} and v_{w_j} by multiple embeddings of Eq. (1):

$$v_{c_j} = \sum_{i=1}^{n} v_{c_{ji}}^T M_i, \quad v_{w_j} = \sum_{i=1}^{n} v_{w_{ji}}^T M_i$$

Here, $v_{w_{ji}}$ is the j^{th} word vector in the vocabulary from the target corpus in the i^{th} embedding space, and $v_{c_{ji}}$ is the j^{th} context vector in i^{th} embedding space. Then our loss function can be rewritten as:

$$\begin{aligned} loss &= \sum_{w_j \in Vocabulary} \left\| \sum_i^n v_{c_{ji}}^T M_i - \sum_i^n v_{w_{ji}}^T M_i \right\|_2 \\ &= \sum_{w_j \in Vocabulary} \left\| \sum_i^n \left(v_{c_{ji}}^T - v_{w_{ji}}^T \right) M_i \right\|_2 \\ &= \left\| \sum_i^n V_{c_i} M_i - \sum_i^n V_{w_i} M_i \right\|_2 \end{aligned} \tag{4}$$

Here, V_{w_i} is the matrix of all target word vectors in the target corpus in the i^{th} embedding space. Also V_{c_i} is the matrix of all context vectors in the target corpus in the i^{th} embedding space.

We want to minimize the loss function of Eq. (4) to minimize the total distance. When we compute one particular transfer matrix M_k, we regard the other transfer matrices as known. According to the Kronecker product identity and Lyapunov equation, we can convert our aim into linear regression.

Suppose

$$\left\| \sum_i^n V_{c_i} M_i - \sum_i^n V_{w_i} M_i \right\|_2 = 0.$$

Then we have:

$$(V_{c_k} - V_{w_k}) M_k = \sum_{i \neq k}^{n} (V_{c_i} - V_{w_i}) M_i$$

According to the Kronecker product identity and Lyapunov equation we obtain:

$$I^T \otimes (V_{c_k} - V_{w_k}) \operatorname{Vec}(M_k) = \operatorname{Vec}\left(\sum_{i \neq k}^{n} (V_{c_i} - V_{w_i}) \cdot M_i \right) \tag{5}$$

Here, I is the identity matrix and $\operatorname{Vec}(X)$ denotes the vectorization of matrix X formed by stacking the columns of X into a single column vector. We can see that Eq. (5) is a classical linear regression problem, thus we can utilize stochastic gradient descent (SGD) to compute the transfer matrix M_k. We compute the other transfer matrices in the same way. After several iterations, we can obtain an optimal result.

The difference between our method and the original word2vec model is that we want to minimize this distance objective function, instead of the product of the predicted probabilities of all the target words. To minimize the objective function, we use stochastic gradient descent (SGD) in computing the transfer matrix. The difference between our objective function and the objective function in Ref. [20] is that our distance is between the target word and context word, while Mikolov's objective function measures the distance between the target word transformed by the matrix and its translated word. Another difference is that the distance in Mikolov's work is used in the target language embedding space. In our work, the distance is not used in any pre-trained embedding space, but in the embedding space trained by the target corpus.

The main difference between our method and the 1 to N+ method [22] is also the loss function. The loss function of Ref. [22] is computed on the global dataset, and the function measures the variation of the words in the pre-trained embedding space and meta-embedding space. This loss function is independent from the target corpus. On the other hand, our loss function measures the word distance in the target corpus like the word2vec model. Our motivation is to optimize embedding results to perform best on the target corpus.

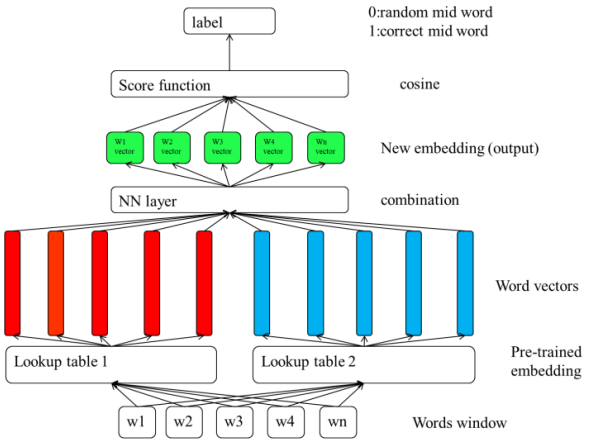


Fig. 6 Neural network configuration.

In our experiments, we use TensorFlow [1] to implement the optimization. We define pre-trained embedding results as the placeholder, which can assign different pre-trained embeddings and define transfer matrices as variables. Then we apply a gradient decent optimizer to minimize the loss function.

3.3 Non-linear Combination

In addition to using transfer matrices, we also try to utilize a neural network to train a better representation of word vectors from pre-trained word vectors. The structure of the neural network is depicted in **Fig. 6**.

The input layer is the word window, on which pre-trained embeddings are mapped to produce word vectors.

The combination layer is a neural network which is allocated with 300 nodes and produces a customized embedding result. We will use this embedding result to compute new word vectors. After computing new word vectors, we apply a loss function that takes the sum of the context word vectors and chooses the word closest to the middle word vector. Here, the loss function is the cosine score between the sum of the context word vectors and the middle word vector.

$$loss = -\frac{V_{context} \cdot V_{mid}}{\|V_{context}\| \cdot \|V_{mid}\|} = -\frac{\sum_{i=1}^{d} (V_{context_i} \cdot V_{mid_i})}{\sqrt{\sum_{i=1}^{d} {V_{context_i}}^2} \cdot \sqrt{\sum_{i=1}^{d} {V_{mid_i}}^2}} \tag{6}$$

Here, $V_{context}$ is a context word vector, V_{mid} is a mid word vector, d is the dimension of these word vectors. Suffix i means the i-th element in a word vector. Notice that, the context word and mid word here are both from the target corpus.

The i-th element comes from the new embedding layer, where the value is computed in the combination layer by the follow formula:

$$V_i = f_{act}\left(\sum_{j=1}^{m} w_j P_{ji} + b \right) \tag{7}$$

Here, V_i is the i-th element in the combined word vector, and m

is the count of pre-trained embeddings. Also, P_{ji} is the i-th elements in j-th pre-trained embedding vector, w_j is a weight on P_{ji}, and b is a bias. Function $f_{act}(\,.\,)$ is the sigmoid function is used as activation.

The most important difference between our non-linear combination and the CBOW model [3] is that our objective function is trained on the target corpus. Pre-trained embedding results are based on general datasets, such as Wikipedia and Google News, which can have disagreements in vocabularies and distributions from the target dataset. While global embedding results were trained based on the objective function over global datasets (e.g., Wikipedia, Google News), our final embedding results were trained over our target dataset. So it is expected that the customized embedding result can fit the target dataset better than pre-trained embedding results.

After we obtain the final target word vectors, we construct a document vector as the sum of all the word vectors in the document. We compute the cosine similarity between each article pair as its semantic similarity by the formula as below:

$$V_D = \sum_{w\ in\ D} V_w$$

$$Semantic\ simlarity(A, B) = \cos(V_{D_A}, V_{D_B})$$

Here V_D is the document vector and V_w is the word vector for each word in D. V_{D_A} and V_{D_B}, respectively, are the document vectors of articles A and B, respectively.

3.4 Word Overlap between Article Pair

Common words in two articles are the most convincing evidence of strong relationship between the two articles. We also utilizing word overlap between article pairs as a feature in our model. We define word overlap between articles A and B as:

$$word\ overlap(A, B) = |WS_A \cap WS_B|$$

Here, WS_A and WS_B are word sets in article A and article B, respectively, where stopwords are removed.

3.5 Jaccard Similarity Coefficient at Article Level and Segment Level

In combination with the word embedding models, we utilize Jaccard similarity coefficient to quantify word overlaps between two articles. The Jaccard similarity coefficient between two articles A and B is defined as below.

$$Jaccard\ similarity\ (A, B) = \frac{|WS_A \cap WS_B|}{|WS_A \cup WS_B|}$$

Here, WS_A and WS_B are word sets in article A and article B, respectively, where stopwords are removed. We do not apply square root to the denominator to reflect overlap sizes. Rather we adopt the above normalized Jaccard similarity. We call this feature as *Jaccard similarity* (*article level*).

Although our goal is to determine whether two articles should be merged or not, we also evaluate Jaccard similarity on every pair of segments of two articles. In this paper, segments correspond to paragraphs of articles. We argue that, when human editors check content overlap of two articles, they closely compare small text fragments, rather than the whole articles. We also expect that the similarity between segments in two articles can be an effective feature for predicting mergeable articles. We compute the Jaccard similarity between every segment pair in two articles, as segment similarities. Then for each article pair, we adopt the similarity coefficient of the most similar segment pair as a feature, called *Jaccard similarity* (*segment level*), which is compute by:

$$Jaccard\ similarity\ (segment\ level) = \max_{i\in I,\ j\in J} (Jaccard\ similarity\ (A_i, B_j)),$$

where A_i and B_j, respectively, are segments in articles A and B, respectively.

3.6 Segment Similarity Distribution

In addition to the most similar segment pair, the whole segments in one article pair can be related to mergeability of article pairs. Content overlaps can be either concentrating on particular segment pairs, or distributed in most of segment pairs of the two articles. In our previous research [22], the distribution of segment similarities is effective for link scope prediction. Thus, we utilize the following statistical features to characterize the distribution of segment similarities: *mean*, *variance*, *standard deviation*, and *coefficient of deviation*, where similarities are measured by Jaccard similarity over all segment pair.

3.7 Common Link-to Article Count

Features from Sections 3.2 to 3.6 for predicting mergeable/merged article pairs are all regarding textual features. A hyper link, or simply a link from one article to another leads readers to related or more detailed information regarding the link origin. Two articles of a pair can share multiple link-to articles. Such commonly linked articles are likely to share common information of the article pair. Thus, we count the number of commonly linked articles from both articles of the pair as a feature named *common link-to article count*, which is expected to indicate relatedness even when word overlaps or semantic similarities are scarce.

3.8 Multimodal Similarity-based Merge Prediction (MSBMP)

Our proposed features consist of: 1. Multiple word embeddings combined by transfer matrix. 2. Jaccard similarity (article level). 3. Jaccard similarity (segment level). 4. Segment similarity distribution, that is a collection of statistical features on segment similarities, consisting of *mean*, *variance*, *standard deviation* and *coefficient of deviation*. 5. Common link-to article count. 6. Word overlap between article pairs. We utilize Random Forest as our classifier by two reasons: 1) Our features show no obvious linear relationship, where tree-based classifier is expected to perform better. 2) Our datasets are imbalanced in the way that mergeable article and merged article pairs are much less than non-mergeable article pairs, while Random Forest is proved to perform better than other popular classifiers over imbalanced datasets, as shown in Refs. [10], [21]. So, we choose Random Forest as our classi-

fier. We call our method as *Multimodal Similarity- Based Merge Prediction* (MSBMP).

3.9 Task 1 Prediction (Mergeable Pairs)

Task 1 is finding articles which are mergeable with a given target Wikipedia article. Our algorithm evaluates candidate articles by confidence score, which is the probability of article pairs being mergeable, estimated by the trained Random Forest classifier. To handle the case where the target article has no mergeable articles, we give a threshold on minimum confidence score to be judged as mergeable. Our algorithm reports articles having confidence score above the threshold as mergeable with the target article. Otherwise, no article is reported as mergeable.

We also rank the candidate articles by the scoring function, and examine the ranking quality by Precision@K, Recall@K, and mean reciprocal rank (MRR). Random Forest classifiers output the probability of each decision, and we use this probability to rank candidate articles.

3.10 Task 2 Prediction (Merged Pairs)

Task 2 is to predict whether a given article pair should be merged or not. Our proposed method produces confidence score on each article pair. In Task 2, the features and classifier are the same as Task 1, but the training dataset is sampled from merged article pairs. Also, we set a tighter threshold on the confidence score than mergeable articles.

4. Benchmark Datasets and Evaluation Schemes

4.1 Reference Dataset Construction

To evaluate our methods for finding mergeable and merged article pairs, we extracted two reference datasets from Wikipedia. For the mergeable article pairs, we randomly extracted 5,460 pairs of articles in total which were suggested to be merged together, from the category page (https://en.wikipedia.org/wiki/Category: All_articles_to_be_merged). These articles in Wikipedia are labeled as "It has been suggested that this article be merged into…". **Table 1** shows where these mergeable articles appear in the ranked result of WikiSearch, in which the title words of the given article are used as search keywords.

The second reference dataset consists of merged article pairs. We extracted 5,000 pairs of articles in total which were actually merged, from the category page (https://en.wikipedia.org/w/index.php?title=Category: Redirects_from_merges). One of the two articles in a merged pair is labeled as "This page is a redirect from a merge…". The difference between these two reference datasets is that the mergeable article pair is currently under discussion, and in the future the pair may be either merged or rejected. On the other hand, the merged article pairs are already merged at the collection time.

We also need to evaluate the situation such that the given target article has no mergeable article. We construct a reference dataset consisting of randomly selected articles which have no mergeable article, and have no article that was merged with. Our criterion for non-mergeable articles is that the article's talk page is not mentioning about merger, and not appearing in the merged articles.

Table 1 Distribution of mergeable articles in ranked results of WikiSearch.

Ranking by WikiSearch	Count of mergeable articles
Top 1-2	1074
Top 3-10	924
Top 11-20	416
Top 21-50	251
Top 51-100	122
Not appearing in WikiSearch top-100 but appearing in common link-to	344
Not appearing in WikiSearch top- 100 and not appearing in common link-to	2329
Total	5460

We call these article pairs as non-mergeable article pairs.

A merged article pair is stricter than a mergeable article pair, because a consensus to merge them was actually reached. Note that merged article pairs are not appearing in the mergeable dataset, because already merged pairs do not appear in WikiSearch results.

We collect candidate articles of a target article by WikiSearch, where the title of the target article is used as query words, and top-100 articles ranked by WikiSearch are collected. Table 1 shows distribution of reference mergeable articles in ranges of ranking by WikiSearch.

We observe that 51 percent of the reference mergeable articles are ranked below 100 or not retrieved by WikiSearch. Therefore, to evaluate performance on candidate articles that are out of WikiSearch top-100, we add more candidate articles through link neighbors. For each target article, we collect articles linked within two-hop distance from the target. For example, suppose that for article A and article B, there are articles C, D and E which have links to both A and B. In this situation, A and B are two-hop neighbors. Also, C, D and E are 1-hop neighbors of A and B. Then we call B has three common link-to articles with A. To reduce candidates by neighbors, we select top-20 articles ranked by the number of common link-to articles with the target, namely articles linked from both the target and a two-hop neighbor.

We call a target article that appears in the mergeable (resp. merged) article pairs as a mergeable (resp. merged) target, and a target article that is neither in the mergeable pairs nor in the merged pairs as a non-mergeable target.

Once an article was merged with the target article, it will no longer appear in any WikiSearch results. Thus, the correct merged article has to be added to the candidates of each merged target.

Since one target may have a quite limited number of true mergeable/merge articles, such as one or two, we restrict the number of candidates to be just 120 or 121, as follows. Each mergeable target has 120 candidate articles (top-100 WikiSearch + top-20 common link-to articles), but whether it has a correct answer in the candidates is unknown. But for merged targets, their candidates consist of 121 candidate articles (top-100 WikiSearch + top-20 common link-to articles + correct answer).

As listed in **Table 2**, we construct four datasets, named as Mergeable-1, Merged-1, 10, 100, where the number $x = 1, 10$, 100 indicates that the ratio of mergeable/merged targets to non-

Table 2 Benchmark datasets.

dataset	target articles	candidate articles
Mergeable-1	5460 mergeable targets+ 5460 non-mergeable targets.	120 candidates for each target.[!] Recall upper-bound is 0.287[!!!]
Merged-1	5000 merged targets+ 5000 non-mergeable targets	121 candidates for each target[!!] Recall upper-bound is 0.5
Merged-10	5000 merged targets+ 50000 non-mergeable targets	121 candidates for each target[!!] Recall upper-bound is 0.09
Merged-100	5000 merged targets+ 500000 non-mergeable targets	121 candidates for each target[!!] Recall upper-bound is 0.0099

[!]: 120 candidates consist of top-100 WikiSearch + top-20 common link-to articles.
[!!]:121 candidates consist top-100 WikiSearch + top-20 common link-to articles +correct answer.
[!!!]: WikiSearch did not retrieve correct articles for 2329 mergeable targets. So (5460-2339) targets have no answer in their candidates.

mergeable targets is 1 to x. We split each dataset into 50 percent for training and 50 percent for test, and adopt two-fold cross validation in computing scores. Regarding splitting, each of mergeable/merged, non-mergeable targets and their candidates are divided into halves, so that each half dataset keeps the same ratio as shown in Table 2. The first half (Part-1) is used for training and the second half (Part-2) is tested. Next, Part-2 is used for training and Part-1 is tested. Then the union of the test results of Part-1 and Part-2 is used to calculate scores for the dataset.

Most of experiments are done on Mergeable-1 and Merged-1. Since non-mergeable candidates do not contain correct answers, they are omitted in calculation of recall. But for FPR and accuracy, the ratio of non-mergeable targets affects the results, thus we vary the ratio of non-mergeable targets from 1 to 100.

4.2 Evaluation Schemes

To evaluate algorithms, we use Precision@K and Recall@K, defined as:

$$Precision@K = \frac{count\ of\ correct\ yes\ answers\ in\ top\ K}{K}$$

$$Recall@K = \frac{count\ of\ correct\ yes\ answers\ in\ top\ K}{count\ of\ real\ mergeable\ articles}$$

Note that most of articles in Wikipedia rarely have more than two mergeable article counter parts, so only one or two correct 'yes' answers are likely to exist. So precision and recall are taking values from limited combinations k'/k, where $0 \le k' \le k$ and k' is a small integer such as 1 or 2. However, by taking macro averages of precision and recall over the whole dataset, we can reveal performance of each algorithm. For the mergeable and merged article datasets, we define:

$$AVGPrecision = \frac{1}{n}\sum_{i=1}^{n} Precision_i$$

$$AVGRecall = \frac{1}{n}\sum_{i=1}^{n} Recall_i$$

Here, n is the number of all the target articles of one dataset, and $Precsion_i$ and $Recall_i$ are those of the i^{th} target article.

In our datasets, non-mergeable targets do not have correct answers in their candidates. Also, if WikiSearch did not retrieve correct answers, they are not included in the candidates. In both cases, we define that the recall for the target is zero. As shown in Tables 1 and 2, Mergeable-1 has 5,460 mergeable targets, but 2,329 targets among those have no correct answer in their candidates. So the upper-bound of AVGRecall of Mergeable-1 is $(5460 - 2329)/(5460 + 5460) = 0.287$. On the other hand, in Merged-1, the candidates of the 5,000 merged targets always contain one correct answer, so the upper-bound of AVGRecall of Merged-1 is $5000/(5000 + 5000) = 0.5$.

To evaluate algorithms for the non-merged situation, we use datasets Merged-1, Merged-10, and Merged-100, and compute Accuracy and False Positive Rate (FPR) to evaluate each method, defined as:

$$Accuracy = \frac{count\ of\ correct\ answers}{count\ of\ total\ mergeable\ pairs}$$

$$FPR = \frac{False\ Postives}{False\ Postives + True\ Negatives} = \frac{incorrect\ 'yes'}{incorrect\ 'yes' + correct\ 'no'}$$

To evaluate from the perspective of where the true mergeable or merged article is ranked in a result ranked by the scoring function of each method, we utilize AVGRecall@K-curve and Mean Reciprocal Rank (MRR). MRR is defined as

$$MRR = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{rank_i}$$

Here, $rank_i$ is the rank of the first correct answer in the result of the i-th article, and n is the number of the articles in the dataset. If the correct answer is not in the ranked result, we set $1/rank_i = 0$. Then the upper-bound of MRR for Mergeable-1 is $(5460 - 2329)/(5460 + 5460) = 0.287$.

5. Experimental Evaluations and Discussions

We conduct a series of experiments to evaluate our methods. The first experiment is comparing single features including combination of multiple pre-trained embeddings. The next experiment is comparing our trained classifier with existing baseline systems. We also discuss optimizing the threshold for the classifier.

5.1 Experiment on Single Methods

In this experiment, we change the ranking parameter K between 1, 2, 3, 4, 5, 10, 20, 50 and 120, to examine correlation between scoring functions and prediction performance. In Table 1, we can see a great majority of mergeable articles are located at rank 20 or higher in WikiSearch results. When K is 1, the algorithm only selects the top-1 article as a possible answer. As most of articles have only one mergeable article, AVGRecall@K and AVGPrecision@K are often close. Hence we just show AVGRecall@K. Baseline algorithms by a single method, consisting of TF-IDF (over the target dataset), Jaccard similarity and simhash,

Table 3 Details of pre-trained embeddings.

Dataset	Word count	Dataset size	Training method
Wikipedia*1	400K vocabulary	6 billion tokens	Glove
Google News*2	3M vocabulary	100 billion tokens	Skip-gram
Common Crawl*3	2.2M vocabulary	840 billion tokens	Glove

*1: https://nlp.stanford.edu/projects/glove/
*2: https://code.google.com/archive/p/word2vec/
*3: https://nlp.stanford.edu/projects/glove/

are also compared on Mergeable-1. For the embedding-based methods, we evaluate three pre-trained embedding results, separately trained by the word2vec model [18] and Glove model [13], as shown in **Table 3**.

Here, we also test embeddings directly trained on Mergeable-1. The similarity of each method is computed as follows:

(1) TF-IDF: TF-IDF vectorization, and cosine similarity.
(2) Jaccard similarity: Binary vectorization, and similarity by Jaccard coefficient.
(3) Simhash: The Simhash algorithm [7].
(4) Embedding: Vectorization by the sum of word vectors, and cosine similarity.
(5) Doc2vec: Doc2vec vectorization, and cosine similarity.

Figure 7 shows the results on AVGRecall@K.

In our algorithm settings, selecting top-K is to reduce low-probability mergeable articles, and the threshold is to give an exact binary answer on each article pair. Here, we draw a curve with changing K to show how AVGRecall@K grows with K. As expected, AVGRecall@K is monotonically increasing with growing K. For Mergeable-1, since 71.3 percent of the target articles do not have true mergeable articles in their candidates, the upper bound of AVGRecall@K is 0.287, not 0.5.

WikiSearch top-2 is an important baseline, because WikiSearch is a readily available tool for editors to find mergeable articles. We expect WikiSearch top-2 will return the target article itself and mergeable article(s). But once two articles were merged, the WikiSearch index is updated, and WikiSearch returns only the target article, thus failing to retrieve the merged article. As shown in Fig. 7, WikiSearch top-1 is much worse than top-2, because WikiSearch returns the target article itself as top-1, but we regard it as a wrong answer.

For evaluation of combining multiple pre-trained embedding results, we compare the sum and concatenation combinations by Garten [9], 1 to N+ combination by Yin [22] and our proposing methods. The results are shown in **Fig. 8**.

We can observe that Transfer matrix combination and Neural network combination perform better than sum and concatenation combinations. The performance of Transfer matrix combination and Neural network combination in this experiment are basically identical. We adopt Transfer matrix combination in the succeeding experiments.

5.2 Experiments on All Features

The combined embedding results above are for comparing semantic similarities. Here, we evaluate Article-level and Segment-level Jaccard similarities for measuring overlaps and duplicates, as well as Segment similarity distribution and Common link-to article count. Datasets Mergeable-1 and Merged-1 are used. Random forest classifiers are trained on all the features, and all-except-one features (ablation test), where a random half of each dataset is used for training and the other half for testing. Confidence score by each classifier is used for ranking the candidate articles. The results of the single features are shown in **Fig. 9** (Mergeable-1) and **Fig. 10** (Merged-1). The results of all features and all-except-one features are shown in **Fig. 11** (Mergeable-1) and **Fig. 12** (Merged-1).

In the single-feature results, Overlap is performing low, but we

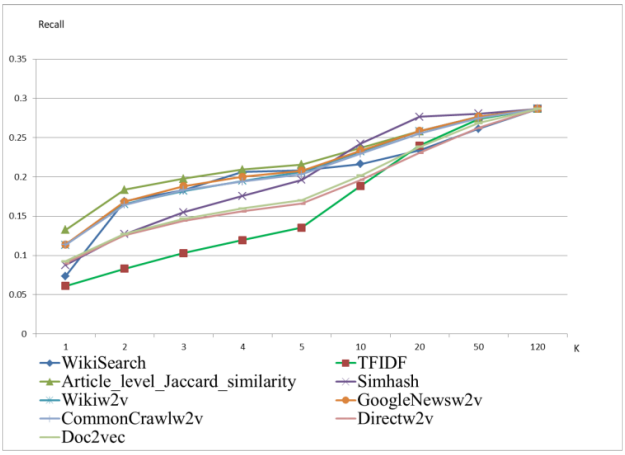


Fig. 7 Results on existing methods (Mergeable-1).

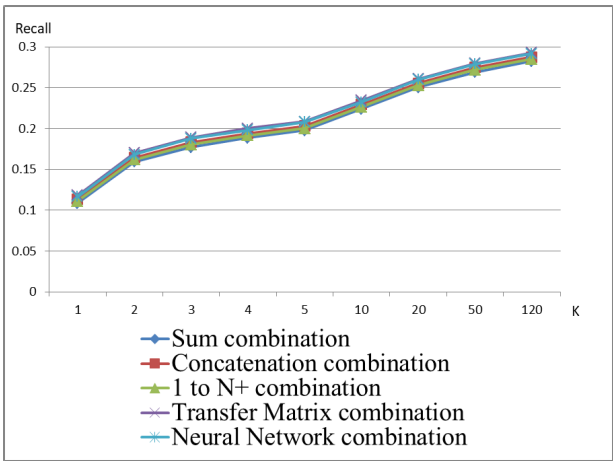


Fig. 8 Results of combining embeddings (Mergeable-1).

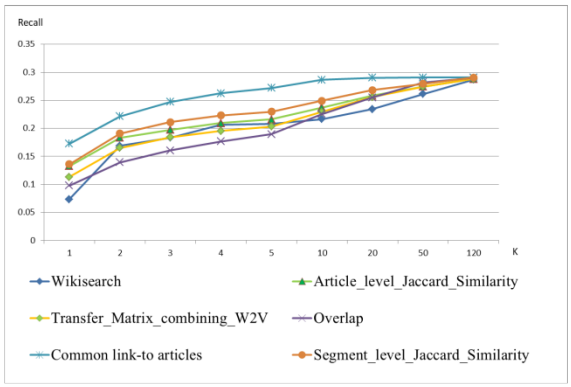


Fig. 9 Performance on single features (Mergeable-1).

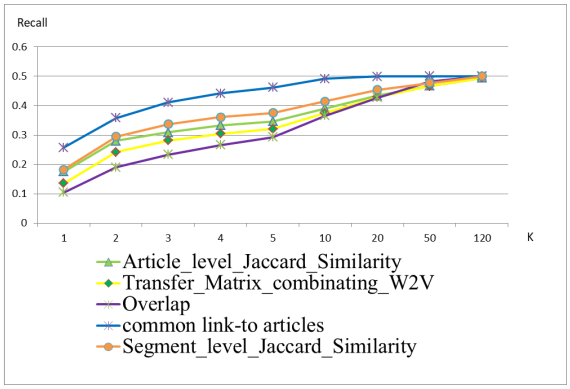


Fig. 10 Performance on single features (Merged-1).

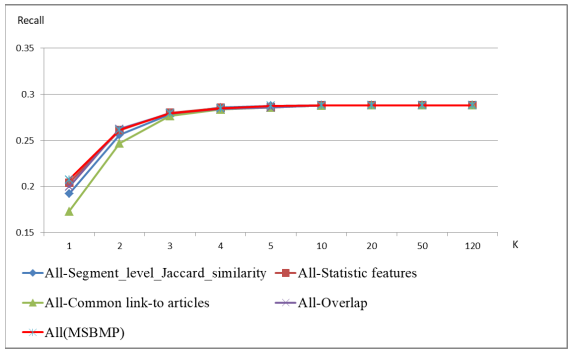


Fig. 11 Performance on all features and remove-one-feature (ablation test) (Mergeable-1).

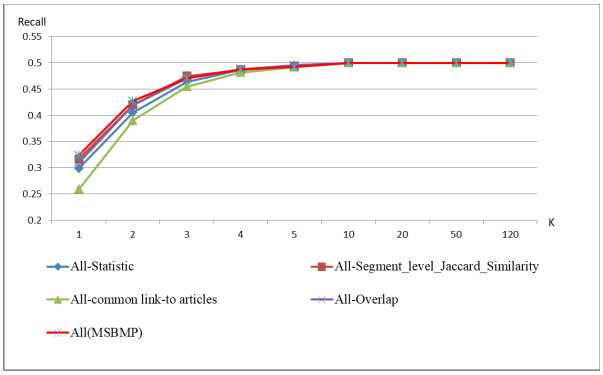


Fig. 12 Performance on all features and remove-one-feature (ablation test) (Merged-1).

expect Overlap can contribute to improve the results when combined with other features. The most effective feature is Common link-to article count and the second one is Segment-level Jaccard similarity. We argue that Common link-to articles is effective because Common link-to article count can capture hidden common topics indicated by links, even when word overlap or semantic overlap is scarce. Keywords related to such hidden topics may not explicitly occur in the target article or candidate articles. On the other hand, links in Wikipedia articles are added by human authors, which can be regarded as an indicator of existence of hidden common topics. Segment-level Jaccard similarity is important because strongly similar text fragments are important clue for article merge.

From Figs. 9 and 10, we find that Segment-level Jaccard similarity performs better than Article-level Jaccard similarity. It can be explained as full articles are long, containing more noisy

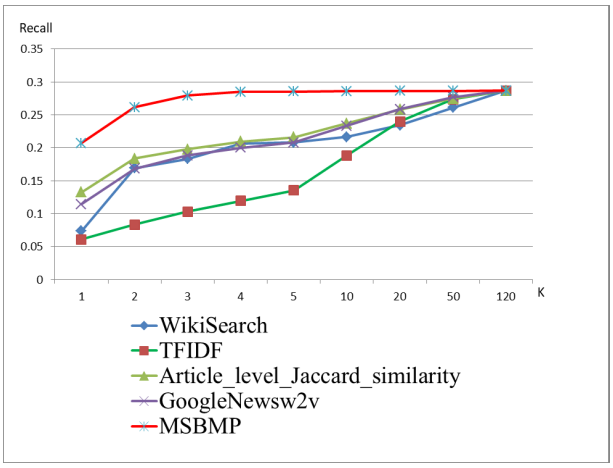


Fig. 13 AVGRecall@K curves by representative methods (Mergeable-1).

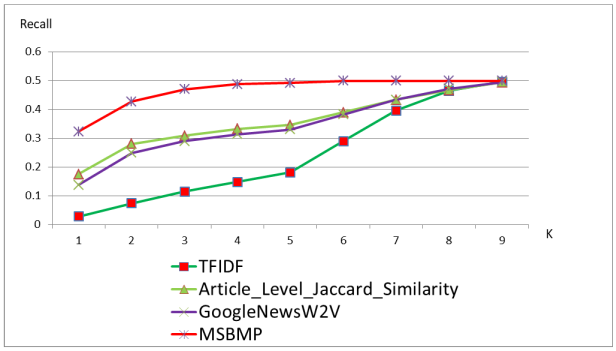


Fig. 14 AVGRecall@K curves on representative methods (Merged-1).

words than the segment level.

We find that the overall best result is the combination of all the features by Random Forest. Therefore, for our proposing method, we choose the combination of the embedding features, Jaccard distance (article level), Overlap, Common link-to article count, Jaccard distance (segment level), and Segment similarity distribution, where Random Forest is used as the classifier.

5.3 Comparison with Existing Methods

In **Fig. 13**, we show the AVGRecall@K curves of representative existing methods, consisting of WikiSearch, TFIDF, Article-level Jaccard similarity, word2vec (Google News), and our proposing method MSBMP, over Mergeable-1. As K grows, AVGRecall of each method also grows, toward the upper limit of 0.287 in dataset Mergeable-1. However, our proposing method rises faster than any other models, closing to upper limit as early as $K = 5$, showing the best performance. Also, our method is significantly better than WikiSearch, meaning that editors can receive significant merits in finding mergeable articles. Similar trends are observed on dataset Merged-1 in **Fig. 14**. As K grows, MSBMP is closing to the upper-bound of 0.5 for Merged-1 faster than any other methods. WikiSearch is not compared on Merged-1, because as explained before, WikiSearch does not find merged articles.

5.4 Experiments on Confidence Thresholds

Figure 15 shows the receiver operating characteristic curves (ROC curves) of the representative methods, where the threshold

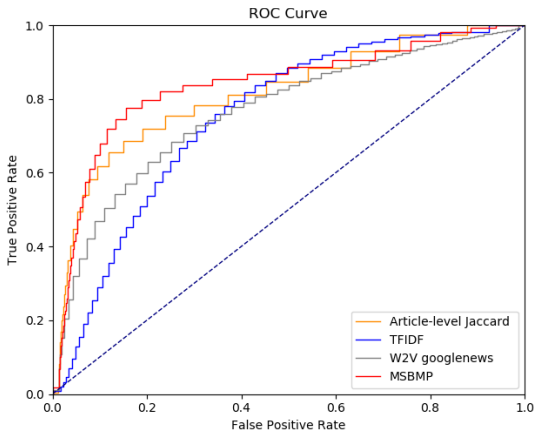


Fig. 15 ROC curves with varying threshold (Mergeable-1).

Table 4 AUC scores (Mergeable-1).

Method	AUC score
TF-IDF	0.745
Jaccard similarity (article level)	0.757
Embedding (Google News)	0.795
MSBMP (proposed)	0.827

Table 5 Precision, Recall and F1 scores (Mergeable-1).

Methods	Precision	Recall	F1
Jaccard similarity (article level)	0.18	0.16	0.17
TFIDF	0.09	0.06	0.07
W2V	0.17	0.15	0.16
MSBMP (proposed)	0.32	0.24	0.27

on confidence score is changed from 0 to 1. We only select the target articles having a mergeable article that can be searched by WikiSearch. This is to allow true positive rate to reach 1.0. We set the maximum value of K as 120, meaning that if the threshold is 0 then all the articles will be classified as mergeable. In this situation, the curve reaches to point $(1, 1)$. If the threshold equals 1, then all the articles will be classified as non-mergeable articles, and the curve reaches to point $(0, 0)$.

In Fig. 15, we can observe that our method is closer to the upper left corner than all the other methods. It means our method is more likely to rank positive articles higher. The upper left corner point of our method is around $(0.2, 0.8)$. At this point the threshold is 0.80, which can be regarded as the optimum threshold for classifying mergeable articles.

Table 4 shows the AUC (area under the ROC curve) score of each method on Mergeable-1.

The results in Table 4 shows that our proposed method MSBMP achieves the highest AUC score, showing the best model performance aggregated over varying threshold.

From the ROC curves, we find the optimum cut-off point for each method, and compute the precision, recall, and F1 score for each method. **Table 5** shows that MSBMP shows the highest F1 score.

To compare the methods in terms of ranking ability, we compute the mean reciprocal rank (MRR) of each method, namely how higher the correct mergeable article is ranked. **Table 6** shows that our proposed method MSBMP's MRR score is highest, indicating that MSBMP performs better than the other methods in ranking ability.

Table 6 Mean reciprocal rank (MRR) results (Mergeable-1).

Method	MRR score
TF-IDF	0.081
Jaccard similarity (article level)	0.162
Embedding (Google News)	0.152
MSBMP (proposed)	0.243
WikiSearch	0.123

Table 7 Accuracy and FPR of MSBMP on merged article pairs.

Dataset	Accuracy	FPR
Merged-1	0.676	0.484
Merged-10	0.590	0.427
Merged-100	0.581	0.418

5.5 Experiments on Handling Non-mergeable Articles

In this experiment, we apply MSBMP to test accuracy and false positive rate (FPR). We use datasets Merged-1, −10, and −100, which are unions of non-mergeable targets with merged targets, where the ratio of the merged targets to non-merged targets is controlled from 1 : 1, 1 : 10, to 1 : 100. In this experiment, we set threshold as 0.8. WikiSearch is not compared in this experiment, because WikiSearch always returns certain articles as positives. **Table 7** shows the accuracy and FPR results of our proposed method.

From Table 7, we find that as the ratio of non-mergeable articles increases, both accuracy and FPR decrease. The degradation of accuracy indicates that our algorithm predicts wrong answers more on non-mergeable article pairs. But even when the ratio of non-mergeable pairs reaches 100 times to merged pairs, both accuracy and FPR are still decreasing slowly, indicating that our algorithm has stable performance for prediction even when merged articles are rare.

5.6 Discussions

From the results, we can find that the results of the word2vec-based methods are superior to TF-IDF, and simhash-based methods. The reason is that while our goal is to find mergeable or merged article pairs, simhash focuses on literal similarity. TF-IDF is influenced by local datasets, performing worse in our datasets. WikiSearch is also utilizing an improved TF-IDF algorithm. So WikiSearch results can be seen as TF-IDF results over the whole Wikipedia. But we find that Article-level Jaccard similarity performs better than TF-IDF and simhash, indicating that measuring overlaps on the whole words is more advantageous than counting matching words or keywords.

We compared the four single embedding methods. The embedding model directly trained only on the target dataset performs worst as we predicted, since the target dataset is much smaller than the other pre-trained models and our combined model. The single embedding result trained over Wikipedia is not the best in the single embedding results. It can be explained as Wikipedia is the smallest corpus in publicly-available pretrained datasets. The combined embedding methods are mostly better than the single embedding results. This is because the combined embedding supports more cases than the single embedding methods, yielding

a high accuracy. Another reason is that combining two different embeddings can reduce excessive bias on the vector generated for each word in the target dataset. The last reason is that our objective function is more adapted to a new particular target dataset, which is not reflected on the embedding results trained over general large corpora.

From Figs. 9 and 10, we can find that Common link-to article count is quite effective. We consider the reason is that links are added by human authors, indicating hidden common topics which are hard to be extracted by word overlaps or semantic similarities. In our experiment, we find Common link-to article count performs especially better in the situation where one article is short (Wikipedia Guidelines 3 and 4).

We show an example on the above case. The articles "Almost Real" and "The Bridges of Madison Country (musical)" are judged to be merged together by editors. But the two articles' Jaccard similarity is only 0.007, and the cosine similarity of these two articles in the embedding space is about 0.5. But within two-hop neighbors, they have three common link-to articles "Jason Robert Brown," "Kelli O'Hara," and "Winterset Iowa," contrasted with other candidates "South Pacific (musical)" and "The Bridges of Madison Country," both have two common link-to articles. This example shows Common link-to article count is finding implicit relatedness that is hard to be found by word overlaps and semantic similarities.

6. Evaluating Impact of Overlap Size and Article Length

To further test our method on various overlaps of article pairs, we test our method on low, middle and high overlaps in article pairs. The low overlap is the pairs that have less than 20 co-occurring words. The middle overlap is the pairs that have between 21 and 60 co-occurring words, and the high overlap is the pairs having more than 60 co-occurring words. In this experiment, we intend to investigate how overlaps affect prediction of each method, through evaluating how well true positives (mergeable targets) are ranked. So in this experiment, we select the 5,460 mergeable target articles and their candidates from Mergeable-1. We call this dataset as Mergeable-1-positive. The overlap distribution of Mergeable-1-positive is shown in **Fig. 16**. The average recall@1 results of the compared methods over the three subsets are shown in **Table 8**.

From the results shown in Table 8, we observe that the overlap size greatly affects the accuracy results of all the considered methods. From Table 8, we find that Jaccard distance performs better in the low overlap size, and the embedding-based methods perform better in high overlap size.

Now let us consider the relationship between article length and overlap size. In contrast to article pairs that are both short, article pairs that are both long are more likely to have high overlap. For the same reason, article pairs that are both short tend to have low overlap. So our principle is that for article pairs that are both long, we will give larger weights on embedding-based methods, and for article pairs that are both short, we will give larger weights on Jaccard distance. But for pairs where one article is short and the other is long, the expected overlap size is not simply proportional to the total article length. So we prefer to put more weight on semantic relatedness.

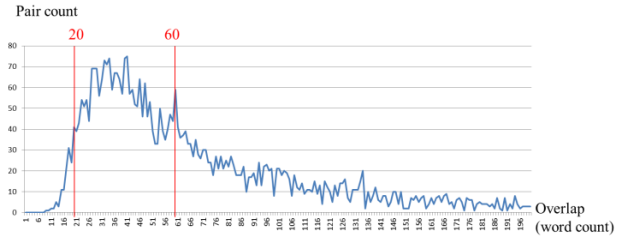


Fig. 16 Distribution of word overlapping (Mergeable-1-positive).

Table 8 AVGRecall@1 results over overlap sizes (Mergeable-1-positive).

Method |/Overlap	[0,20]	(20,60]	(60,+∞]
mergeable pair count	192	2204	3064
Jaccard (article level)	**0.144**	**0.184**	0.148
Embedding (Wikipedia)	0.067	0.122	0.230
Embedding (Google News)	0.067	0.129	0.227
Embedding (Common Crawl)	0.071	0.126	0.225
Transfer matrix combination (Google News + Common Crawl)	0.065	0130	0.229
Transfer matrix combination (Common Crawl + Google News)+ Jaccard(article level)	0.098	0.154	0.231
MSBMP (proposed)	0.106	0.172	**0.252**

In Table 8, we find that the Jaccard distance performs best in the low overlap pairs, while the embedding-based methods perform better in the high overlap pairs. That can be explained as the article pairs with low overlap are relatively short, so Jaccard distance is more effective. The embedding-based methods perform better on pairs having high overlap, because these article pairs are likely to be longer, so semantic similarities of non-overlapping parts give more information.

We also try to determine the optimum weight between Jaccard distance and semantic relatedness, by the overlap size and article length. But the results are not improved. It appears that this parameter optimization requires further studies. We design an experiment to see how they perform on article groups of different relatedness. We collect three groups as follows:

a. Merge group consists of 100 pairs of mergeable articles.

b. Related group consists of 100 pairs of related articles, where related articles are such that we search articles by an article title on WikiSearch, and the article of the title and one of the search results is given as a related article pair.

c. Random group consists of 100 pairs of random articles.

We compute Jaccard distance of each article pair, on the segment and article levels. For the article level, we choose the shortest Jaccard distance between all the segment pairs of one article pair. Then we rank the pairs by the descending order of Jaccard distance. **Figures 17** and **18**, respectively, show the trends of Jaccard distance on the article level and segment level, respectively.

Figure 17 shows that the curves of the three groups are totally

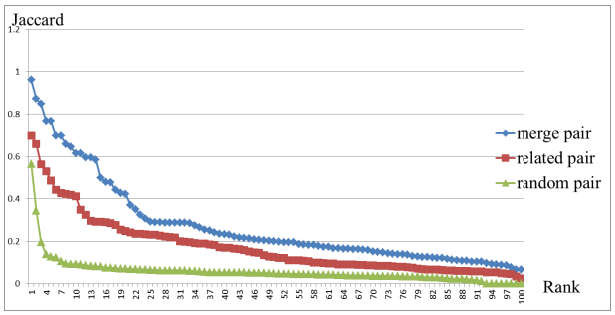


Fig. 17 Jaccard similarity on article level (Mergeable-1).

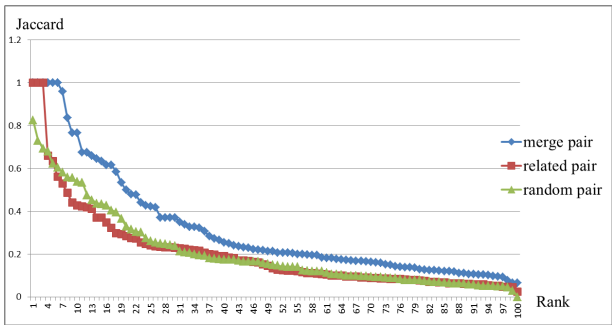


Fig. 18 Jaccard similarity on segment level (Mergeable-1).

separated from each other. In Fig. 17, the curves of related pairs and random pairs interweave with each other, but the curve of merged pairs is distinctively higher than the other two curves. This indicates that the Jaccard distance on the segment level is separating mergeable pairs better than the article level, in this classification problem.

7. Conclusion

In this paper, we proposed Multimodal Similarity-Based Merge Prediction (MSBMP) for detection of mergeable and merged article pairs. MSBMP integrates textual and semantic similarities as well as link-structural similarities, to deal with a variety of situations where article merge is necessary. Regarding linear and non-linear combination methods of multiple embedding results for semantic similarities, we discussed the differences between pre-trained large datasets and target dataset, and introduced a new objective function. This objective function can train a model more fitted to a particular target dataset.

In addition to semantic similarities, we also considered features on word overlap and semantic similarity between article pairs. We further discussed combining embedding methods and Jaccard similarity to cope with both semantic similarity and word overlaps. Rigorous evaluations on real mergeable and merged datasets shows that combination of features on embeddings, Jaccard similarity (article- and segment- levels), word overlap, Common link-to article count, and Segment similarity distribution achieves around 5 percent improvement on Recall@1 over WikiSearch. We showed Jaccard similarity (segment level) and Segment similarity distribution are indispensable features. We also found that Common link-to article count is a particularly strong feature when one article is short, and Jaccard similarity is especially effective on highly similar segments.

Acknowledgments This work was in part supported by JSPS KAKENHI Grant Number 19K11983. The authors are grateful for the helpful and constructive comments by the reviewers and editor.

References

[1] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M. and Kudlur, M.: TensorFlow: A System for Large-Scale Machine Learning, *OSDI*, Vol.16, pp.265–283 (2016).

[2] Alonso, O., Fetterly, D. and Manasse, M: Duplicate news story detection revisited, *Asia Information Retrieval Symposium*, pp.203–214, Springer, Berlin, Heidelberg (2013).

[3] Bengio, Y., Ducharme, R., Vincent, P. and Jauvin, C.: A neural probabilistic language model, *Journal of Machine Learning Research*, Vol.3, pp.1137–1155 (2003).

[4] Breiman, L.: Random forests, *Machine Learning*, Vol.45, No.1, pp.5–32 (2001).

[5] Broder, A.Z.: On the resemblance and containment of documents, *Proc. Compression and Complexity of Sequences 1997*, pp.21–29, IEEE (1997).

[6] Broder, A.Z., Glassman, S.C., Manasse, M.S. and Zweig, G.: Syntactic clustering of the web, *Computer Networks and ISDN Systems*, Vol.29, No.8-13, pp.1157–1166 (1997).

[7] Charikar, M.S.: Similarity estimation techniques from rounding algorithms, *Proc. 34th ACM Symp. Theory of Computing*, pp.380–388 (2002).

[8] Ferrero, J., Agnes, F., Besacier, L. and Schwab, D.: Using Word Embedding for Cross-Language Plagiarism Detection, *EACL 2017*, p.415 (2017).

[9] Garten, J., Sagae, K., Ustun, V. and Dehghani, M.: Combining distributed vector representations for words, *Proc. 1st Workshop on Vector Space Modeling for Natural Language Processing*, pp.95–101 (2015).

[10] Hong, X., Chen, S. and Harris, C.J.: A kernel-based two-class classifier for imbalanced data sets, *IEEE Trans. Neural Networks*, Vol.18, No.1, pp.28–41 (2007).

[11] Ng, A.: Sparse autoencoder, *CS294A Lecture Notes*, Vol.72, pp.1–19 (2011).

[12] Pan, S.J. and Yang, Q.: A survey on transfer learning, *IEEE Trans. Knowledge and Data Engineering*, Vol.22, No.10, pp.1345–1359 (2010).

[13] Pennington, J., Socher, R. and Manning, C.: Glove: Global vectors for word representation, *Proc. 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp.1532–1543 (2014).

[14] Peng, H., Li, J., Song, Y. and Liu, Y.: Incrementally Learning the Hierarchical Softmax Function for Neural Language Models, *AAAI* pp.3267–3273 (2017).

[15] Pertile, S.D.L., Moreira, V.P. and Rosso, P.: Comparing and combining Content-and Citation-based approaches for plagiarism detection, *Journal of the Association for Information Science and Technology*, Vol.67, No.10, pp.2511–2526 (2016).

[16] Schleimer, S., Wilkerson, D.S. and Aiken, A.: Winnowing: Local algorithms for document fingerprinting, *Proc. 2003 ACM SIGMOD Int. Conf. Management of Data*, pp.76–85 (2003).

[17] Schofield, A., Thompson, L. and Mimno, D.: Quantifying the effects of text duplication on semantic models, *Proc. 2017 Conference on Empirical Methods in Natural Language Processing*, pp.2737—2747 (2017).

[18] Mikolov, T., Ilya, S., Kai, C., Greg, C. and Jeffrey, D.: Distributed Representations of Words and Phrases and their Compositionality, *NIPS '13*, pp.3111–3119 (2013).

[19] Mikolov, T., Kai, C., Greg, C. and Jeffrey, D.: Efficient Estimation of Word Representations in Vector Space, *ICLR '13 Proc. Workshop at International Conference on Learning Representations* (2013).

[20] Mikolov, T., Le, Q.V. and Sutskever, H.: Exploiting similarities among languages for machine translation, arXiv preprint arXiv:1309.4168 (2013).

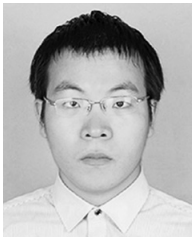
[21] Sun, Y., Wong, A.K. and Kamel, M.S.: Classification of imbalanced data: A review, *International Journal of Pattern Recognition and Artificial Intelligence*, Vol.23, No.4, pp.687–719 (2009).

[22] Wang, R. and Iwaihara, M.: Estimating Reference Scopes of Wikipedia Article Inner-links, *IPSJ Trans. Databases*, Vol.11, No.2, pp.1–9 (2018).

[23] Weissman, S., Ayhan, S., Bradley, J. and Lin, J.: Identifying duplicate and contradictory information in wikipedia, *Proc. 15th ACM/IEEE-CS Joint Conference on Digital Libraries*, pp.57–60, ACM (2015).

[24] Yin, W. and Schütze, H.: Learning meta-embeddings, *ACL 2016*, pp.1351–1360 (2016).

[25] Zhang, Y., Roller, S. and Wallace, B.C.: MGNC-CNN: A Simple Approach to Exploiting Multiple Word Embeddings for Sentence Classification, *Proc. NAACL-HLT*, pp.1522–1527 (2016).
[26] Wikipedia:Merging, available from ⟨https://en.wikipedia.org/wiki/Wikipedia:Merging#Reasons for merger⟩.



Renzhi Wang received his B.S. degree in Computer Science and Technology from Sichuan University in 2013. He received his M.Eng. degree from Waseda University in 2014. He is now a Ph.D. candidate in Waseda University.

Mizuho Iwaihara received his B.Eng., M.Eng. and D.Eng. degrees all from Kyushu University, in 1988, 1990, and 1993 respectively. He was a research associate and then an associate professor in Kyushu University, from 1993 to 2001. From 2001 to 2009, he was an associate professor at Department of Social Informatics, Kyoto University. Since 2009, he is a professor at Graduate School of IPS, Waseda University. He is a member of IEICE, IPSJ, ACM and IEEE CS.

(Editor in Charge: *Sumio Fujita*)