Article

Estimation of cross-lingual news similarities using text-mining methods

Zhouhao Wang [1]*, Enda Liu [1], Hiroki Sakaji [1]*, Tomoki Ito [1], Kiyoshi Izumi [1]*, Kota Tsubouchi [2] and Tatsuo Yamashita [3]

[1] Izumi lab, Department of System Innovation, Graduate School of Engineering, The University of Tokyo; 113-0033 Tōkyō-to, Bunkyō-ku, Hongō, 7 Chome-3-1, Engineering Building No.8; izumi-lab@socsim.org; Tel.: +81-03-5841-6993

[2] Yahoo! Japan Research; ktsubouc@yahoo-corp.jp

[3] Yahoo! Japan Research; tayamash@yahoo-corp.jp

* Correspondence: wangzhouhao94@gmail.com, sakaji@sys.t.u-tokyo.ac.jp,izumi@sys.t.u-tokyo.ac.jp

Academic Editor: name
Version January 23, 2018 submitted to MDPI

Abstract: In this research, two estimation algorithms for extracting cross-lingual news pairs based on machine learning from financial news articles have been proposed. Every second, innumerable text data, including all kinds news, reports, messages, reviews, comments, and tweets have been generated on the Internet, which is written not only in English but also in other languages such as Chinese, Japanese, French and so on. By taking advantage of multi-lingual text resources provided by the Thomson Reuters News, we developed two estimation algorithms for extracting cross-lingual news pairs from multilingual text resources. In our first method, we propose a novel structure that uses the word information and the machine learning method effectively in this task. Simultaneously, we developed a bidirectional LSTM based method to calculate cross-lingual semantic text similarity for long text and short text respectively. Thus, when an important news article is published, users can read similar news articles that are written in their native language using our method.

Keywords: Text Similarity; Text Mining; Machine Learning; SVM; Neural Network; LSTM

1. Introduction

Text similarity, as its name suggests, refers to how similar the given text query is similar to the others, where we normally tend to consider mainly on their semantic characteristics, that is, how close (i.e. similar) their meanings are. Here, the text could be in the form of character level, word level, sentence level, paragraph level, or even longer, document level. In this paper, we mainly discuss the text with the form of sentences (i.e. short text) and documents (i.e. long text).

The objective of this research could be summarized in three key points. The fundamental objective is to develop algorithms for estimation of semantic similarity for the given two pieces of text written in different languages, applicable for both long text and short text, by taking advantage the untapped vast of text resources from Thomson Reuters economics news reports. Secondly, as a practical application and a verification of our model, we are aiming at developing a cross-lingual recommendation system and test benchmark, where it could provide several most-related (for example, 10 results) pieces of Japanese or English text when given an English (or Japanese) article. Thirdly, we excavate cross-lingual resources from the enormous database of Thomson Reuters News and build an effective cross-lingual system by taking advantage of this un-developed treasure.

2. Related Work and Theories

In spite of the length of the text, most of the state-of-the-art methods implemented based on the word embedding methods recently and thus we discuss it in detail in a separate section. To solve semantic text similarity problems, one of the most typical and inspiring methods is Siamese LSTM structure, which is considered as both basis and a competitive baseline of this research.

2.1. Embedding Techniques for Words and Documents

Word embedding techniques, also known as distributed word representation, is one of the most basic concepts and application prevalent nowadays. Word embedding could be further extended to perform on even documents. The embedding techniques capturing both the semantic and syntactic information and converting them into meaningful feature vectors which help to train accurate models for natural language processing (NLP) tasks [1].

Word embedding can be implemented for both monolingual and multilingual task. There are several successful papers working on the monolingual word embedding such as the continuous bag of words models and skip-gram models [2], monolingual document embedding such as doc2vec [3], cross-lingual word embedding [4], as well as cross-lingual document embedding model such as Bilingual Bag-of-Words without Word Alignments (BilBOWA) [5]. Through embedding model, each word, phrase or document would be converted into a fixed length vector representation, where the similarity between each of two words, phrases, or documents could be derived by calculating the cosine distance of their vector representations. Methods are distinctly different for the text data with different length when solving text similarity problem[3]. In respective with the length of the text, textual similarity task could be further categorized into two sub-tasks. Prevalent methods for cross-lingual document (i.e. long text) similarity could be categorized into four aspects [6], Dictionary-based approaches[7], Probabilistic topic model based approaches[8], Matrix factorization based approaches[9], and Monolingual approaches.

2.2. Text Similarities Using Siamese LSTM

A neural network based Siamese recurrent architectures are recently proved to be one of the most effective ways for learning semantic text similarity on the sentence level. Mueller, in his work, implements a Siamese recurrent structure called Manhattan LSTM (MaLSTM) [10], which is practically used as the estimation of relativeness (i.e. similarity) when given any two sentences in English. This structure using Long Short-Term Memory (LSTM)[11] have state-of-the-art performance on both semantic relatednesses scoring task and entailment classification using the SICK database, one of the NLP challenges provided by SemEval[12]. This model could identify how two sentences are similar to each other by trying to “understand" their true meaning on a deeper aspect, like the sentence pairs “He is smart" and “A truly wise man" as the figure demonstrates. They have no common word with different length, but they are indeed highly relevant to each other in terms of their implications, which a human cannot recognize without more consideration and logical analysis, suggesting the difficulty of this challenge.

In our work, we developed a new recurrent structure inspired by MaLSTM, by modifying the Siamese (i.e. symmetric) LSTM modules to an “unbalanced" ones, and add a full-connect neural network layer following the output of LSTM modules, which is more flexible and effective over text similarity task.

3. Methods for Extracting Cross-lingual News Pairs

In this section, we will introduce all fundamental and necessary methods applied in our research. There are mainly three aspects to be elaborated, including methods we applied regarding to the foundation of natural language processing, such as word embedding and TF-IDF. The we explained two applied methods, one of which is classical methods learning, SVM (Support Vector Machine). The other one is neural network method, LSTM(Long-Short Term Memory).

3.1. Distribution Representation

The most traditional and naive way to consider words as features is to treat words as discrete symbols or numbers. This results in a discrete representation of each word and hinders the establishment of relations among these features. In contrast, vector space models consider (embedded) words in a continuous vector space, in which words with similar meanings are separated by small distances. There are two main categories for continuous word embedding: count-based (such as latent semantic analysis) models and predictive-based methods (such as neural probabilistic language models). The count-based models focus on the co-occurrence of the considered word and its neighboring words, whereas the predictive-based models predict a word based on its neighbors using embedding vectors [13]. In this research, we implement a predictive-based model that is known as word2vec; it is based on the skip-gram or continuous bag-of-words model [2].

We train each word from the training text sequence $w_1, w_2, w_3, ..., w_T$ to maximize the objective function

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log p(w_{t+j}|w_t) \tag{1}$$

wherein c is the so-called "window size," which determines how much context information is to be considered for each of the training words. More specifically, we define $p(w_{t+j}|w_t)$ using a softmax function:

$$p(w_O|w_I) = \frac{\exp({v_{w_O}}^T v_{w_I})}{\sum_{w=1}^{W}\exp({v_w}^T v_{w_I})} \tag{2}$$

wherein W is the size of the vocabulary (i.e., the number of disparate words to be considered), and v is the vector representations for either the word w, the input word w_I, or the output word w_O.

However, the calculation of equation2 2 is impractical because the computational cost for calculating the gradient of $\log p(w_{t+j}|w_t)$ is proportional to W, which consists as many as 10^5 to 10^7 terms. In practical, to train the model (i.e., optimize the cost function) in a more computationally efficient manner, we use Noise Contrastive Estimation for approximation during training, as described in [2].

Finally, vector representations with fixed dimension (e.g., 200) can be extracted from the trained model. These word vectors have some outstanding attributes. Because we train our model for each word using its neighboring words, and words with similar meaning usually tend to have similar context, we can calculate the similarity among words using the cosine distance.

3.2. Term Frequency-Inversed Document Frequency (TF-IDF)

TF-IDF is one of the classical weighting models for words, which uses text representations. It is widely used in the natural language processing domain wherein it is commonly applied for weighting words or document features, such as in one-hot bag-of-words representation. The term frequency stands for the number of times a considered word occurs in a specific document, while the document frequency is the number of documents in the corpus that include the word. The inverse document frequency term for a specific word can be expressed as

$$idf = \log\frac{N}{1+df} \tag{3}$$

wherein N is the total number of documents in the corpus. Combining these two concepts, the TF-IDF weight is the product of the TF and the IDF. This scheme loses semantic information for words; thus, so it usually cannot achieve satisfactory performance. However, it measures the weights and importance of each word inside documents and among other documents according to a reasonable definition. In this study, we apply TF-IDF to weight words during document embedding.

3.3. TF-IDF Weighting for Word Vectors

Although there are several ways to form vector representations for documents (i.e., document embedding), we have experimentally discovered that the most effective strategy is to use the TF-IDF weighted sum of the word vectors that are present in each document as features. First, we calculate two TF-IDF weighting models, namely TF-IDF$_{jp}$ and TF-IDF$_{en}$, for each word from English training documents and Japanese training documents. Second, for each Japanese document, the weighted sum document representation can be derived as

$$\mathbf{J_i} = \sum_{m=0}^{N_i} t_{i,m} \cdot \mathbf{w_{i,m}} \tag{4}$$

wherein N_i refers to the number of words in this Japanese document (i.e., Japanese document i), and $t_{i,m}$ stands for the Japanese TF-IDF weight for the m-th word in document i with respect to the considered word. The final term $w_{i,m}$ is the word vector of the m-th word in document i, that is, the vector representation for this considered word.

We apply the same weighting scheme to the English documents. The vector representation for English document i can be expressed as

$$\mathbf{E_i} = \sum_{m=0}^{N_i} t_{i,m} \cdot \mathbf{w_{i,m}} \tag{5}$$

wherein all the definitions of the above variables are the same as those in the Japanese processing case, except the texts are in English.

3.4. Feature Engineering

The selection of features is possibly the most significant and tricky step, in particular, for classical machine learning algorithms such as SVM. This is called "feature engineering" because sometimes the choice of features can greatly affect the results. Fortunately, as one of the most exciting results in this research, we discover that satisfactory results can be generated using the joint cross-lingual document vector that is based on TF-IDF weighted word2vec as a training feature for the SVM model. Although both SVM and TF-IDF weighted word vectors are common in the text mining domain, to the best of our knowledge, this is the first time that the effectiveness of using joint cross-lingual text feature vectors as input for SVM on the cross-lingual text similarity problem has been proved.

More specifically, for the vector representation of a Japanese document $\mathbf{J_i}$ and an English document $\mathbf{E_j}$, the joint feature are defined by

$$\mathbf{f_{i,j}} = (\mathbf{J_i}, \mathbf{E_j}) \tag{6}$$

Via feature engineering, we prepare our training datasets S, which contain a subset S_1 of instances for which the similarity scores are all equal to 1:

$$S_1 = \{(\mathbf{f_{1,1}}, 1), ..., (\mathbf{f_{N,N}}, 1)\} \tag{7}$$

and another subset S_0 of instances for which the similarity scores are all equal to 0:

$$S_0 = \{(\mathbf{f_{1,o}}, 0), ...(\mathbf{f_{Q,P}}, 0)\} \tag{8}$$

wherein N is the total number of cross-lingual training pairs with similarity of 1 (i.e., similar pairs) for training and o is an arbitrary number that belongs to (1, N) and is not equal to 1, such that $\mathbf{f_{1,o}}$ is the set of dissimilar pairs with similarity of 0 (i.e., the pairs are totally unrelated). Moreover, note that $Q, P \in (1, N)$ and $Q \neq N$.

Hence, our final training data S is

$$S = S_1 \cup S_0 \tag{9}$$

3.5. The SVM-based method

SVM is one of the most popular methods for solving both classification and regression tasks. It is originally purposed in 1990s and gradually proved to be effective in many fields including Natural language processing (NLP), pattern recognition and so on [14][15][16].TF-IDF and SVM are useful for tasks in the field of natural language processing. Therefore, we employ TF-IDF and SVM in our method as core technologies. Additionally, we propose a novel structure that uses TF-IDF and SVM effectively for this task. An overview of the structure is illustrated in Figure 1.

The system mainly contains three processing models. As our our training datasets S only contains the data with label 0 or 1, the classification training objective of SVM is very similar to classification using Triplet Loss, which is proved quite effective in embedding and classification tasks.[17] The training procedures normally include the following steps:

1. Use the cross-lingual training data in the form of pre-trained word vectors as input, which is discussed in detail in section 3.1.
2. Weight the word vectors for each of language models using TF-IDF, as introduced in section 3.2 and section 3.3.
3. Train the proposed model using SVM with Platt's probability estimation for the connected cross-lingual document features, each of which are the naive join of two weighted word sum vectors in English and Japanese. This is explained in section 3.4.

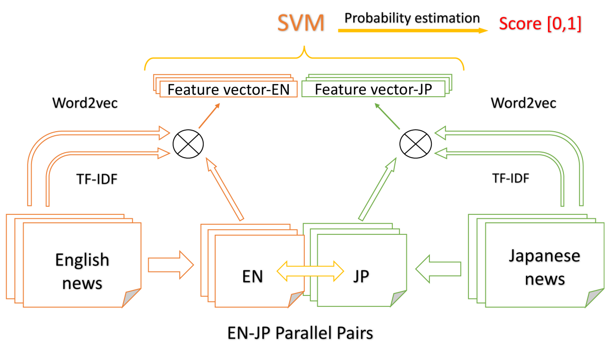


Figure 1. Illustration of our SVM-based method

3.6. A Bidirectional LSTM Based Method

We implement the two independent modules of bi-directional LSTM recurrent neural networks on both English input and Japanese input respectively and the overview of this structure is shown in the figure 2. We use the cross-lingual training data in the form of pre-trained word vectors as input. Feed the word vector sequentially to LSTM modules. This is discussed in detail in the section 3.1. Furthermore, as a limitation of our LSTM modules, we have uniform length of data as input, denoted as "maxlen". The residue of the parts of sequence longer than maxlen will be abandoned, while those input sequence shorter than "maxlen" will be padded with a predefined value (i.e. a word) such as "null" at the tail so that all the input data could be same in length. The two bi-LSTM modules are responsible for English sequence and Japanese Sequence respectively. They generate four hidden layer outputs and we concatenate them into a joint feature. Details is elaborated in the 3.6.1. The joined feature is further fed into a densely-connected neural network of 1 depth, resulting in 1 dimension output $y \in [0,1]$ as the final similarity score of the two inputs cross-lingual data, by means of regression. In general, LSTM-based model pay more attention on the order information of the input sequence, which might significant determine the real meaning of the a sentence written in natural languages.

3.6.1. The bi-LSTM Layer

In this research, we take advantage of bi-LSTM (bi-directional long short-term memory), to enhance the ordinary RNN performance considering both forward and backward information and solve the problem of the long-term dependencies. The updates rules of LSTM for each sequential input $x_1, x_2, ..., x_t, ..., x_T$ could be express as:

$$i_t = \text{sigmoid}(W_i x_t + U_i h_{t-1} + b_i) \tag{10}$$

$$f_t = \text{sigmoid}(W_f x_t + U_f h_{t-1} + b_f) \tag{11}$$

$$\tilde{c_t} = \tanh(W_c x_t + U_c h_{t-1} + b_c) \tag{12}$$

$$c_t = i_t \odot \tilde{c_t} + f_t \odot c_{t-1} \tag{13}$$

$$o_t = \text{sigmoid}(W_o x_t + U_o h_{t-1} + b_o) \tag{14}$$

$$h_t = o_t \odot \tanh(c_t) \tag{15}$$

where h_{t-1} is the hidden layer value of the previous states and the sigmoid and tanh functions in the above equations are also used as activation functions:

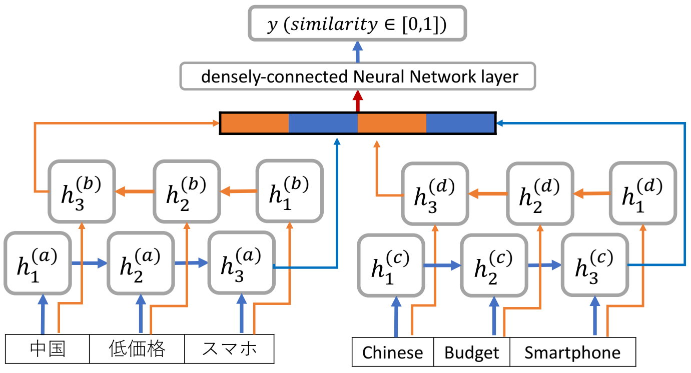
$$\text{sigmoid}(x) = \frac{1}{1+\exp(-x)} \tag{16}$$

$$\tanh(x) = \frac{2}{1+\exp(-2x)} - 1 \tag{17}$$

The weights (i.e. parameters) we need to train include $W_i, W_f, W_c, W_o, U_i, U_f, U_c, U_o$ and bias vectors b_i, b_f, b_c, b_o. A more thorough exposition of the LSTM model and its variants is provided by Graves (2012)[18] and Greff et al. (2015)[19]. In this layer, we use the cross-lingual training data in the form of pre-trained word vectors as input, which is discussed in detail in section 3.1. There are four LSTM modules, constructing two bi-LSTM structures, where we only consider the final output (i.e. final value of the hidden layer) of each LSTM modules: LSTM-a read Japanese text forwardly. The value of hidden layer is denoted as $\mathbf{h_i^{(a)}}$ where i is the i-th input of the sequence, while LSTM-b read backwardly denoted as $\mathbf{h_i^{(b)}}$. Symmetrically, LSTM-c and LSTM-d are used to read English text, denoted as $\mathbf{h_i^{(c)}}$ and $\mathbf{h_i^{(d)}}$. As the results, we obtain four feature vectors derived from hidden layer values of the four LSTM modules, keeping all necessary information regarding to the cross-lingual inputs. We then merge these four features by concatenating them directly:

$$\mathbf{x_{i,j}} = (\mathbf{h_L^{(a)}}, \mathbf{h_L^{(b)}}, \mathbf{h_L^{(c)}}, \mathbf{h_L^{(d)}}) \tag{18}$$

where i and j refer to the document number of the input text for Japanese and English respectively, and vector $\mathbf{h_L^{(a,b,c,d)}}$ refers to the final status (i.e. the value) of the hidden layers of the LSTM module after feeding the last (or the first word, if backwardly) word.



y (similarity ∈ [0,1])
densely-connected Neural Network layer
$h_1^{(a)}$ $h_2^{(a)}$ $h_3^{(a)}$
$h_1^{(b)}$ $h_2^{(b)}$ $h_3^{(b)}$
$h_1^{(c)}$ $h_2^{(c)}$ $h_3^{(c)}$
$h_1^{(d)}$ $h_2^{(d)}$ $h_3^{(d)}$
中国
低価格
スマホ
Chinese
Budget
Smartphone

Figure 2. Illustration of LSTM-based method

161 ### 3.6.2. Dense Layer

We use the most basic component of basic full-dense Neural Network layer as the top layer. The function of this layer could be expressed as:

$$y_{i,j} = f(\mathbf{w}^T \mathbf{x_{i,j}} + b) \tag{19}$$

Here, the function f is also known as "activation" function, and b is the one dimension bias for the neural network and $\mathbf{w}$ is the weight (i.e. the parameters to be trained) of the neural network. In this project, we mainly apply the softplus [20] function as activation function in the dense layer :

$$f(x) = \ln[1 + \exp(x)] \tag{20}$$

As for the optimization, although we are handling a classification problem, based on the experiment results, we find that, instead of using ordinary cross-entropy cost, it performs better if we use Quadratic cost (i.e. mean square error) as the cost function, which could be described as:

$$C = \sum_{v=1}^{N} (y_{true,v} - y_{pred,v})^2 \tag{21}$$

where N is the total number of the training data, while $y_{true,v}$ and $y_{pred,v}$ refer to the true similarity and the predicted similarity respectively. In practice, the stochastic gradient descent (SGD) is implemented by means of the back-propagation scheme. After computing the outputs and errors based on the cost function J, which is usually equal to the negative log of the maximum likelihood function, we update parameters by the gradient descent method, expressed as:

$$\mathbf{w} \leftarrow \mathbf{w} - \varepsilon \nabla_w J(\mathbf{w}) \tag{22}$$

where ε is known as "learning rate", defining the update speed of the hyper-parameters $\mathbf{w}$. However, the training process might fail due to either improper initialization regarding weights or the improper learning rate value set. Practically, based on the results of the experiments, the best performance is achieved by applying the Adam optimizer [21] to perform the parameter updates.

166 ## 4. Experiments and Results

167 ### *4.1. Evaluation Methods*

We use mainly two categories of the evaluations, TOP-N benchmark based on ranks, and traditional criteria for classification such as precision, recall as well as the F1-value. As the applications of this project is to suggest the users several cross-lingual (For instance, English) alternatives news when the user provides a Japanese article as a query, we make the system pick up 1, 5 and 10 of the most similar Japanese alternatives during the evaluation process. The figure 3 illustrates the relationship and evaluation procedures for ranks, TOP-N index.For a given Japanese text (i.e.the query) J_x, calculate the similarity score between J_x and all English text of test data sets $(E_1, E_2, ..., E_x, ..., E_M)$ to derive a list of scores $L_x = (S_{x,1}, S_{x,2}, ..., S_{x,x}, ..., S_{x,M})$, where the corner mark M is the total number of English documents to be considered, and E_x is the true similar article for with similarity score of 1. Then sort this list in the order from large to small and find out the rank (i.e. position, index) of the score $S_{x,x}$ inside this sorted list noted as R_x, the rank for the query document J_x. Repeat this process recursively for N Japanese articles $(J_1, J_2, ..., J_N)$, result in a list of ranks $R = (R_1, R_2, ..., R_N)$ regarding the collections of J_x. Then we take the number of query documents with ranks smaller than N as TOP-N. In other words, TOP-1 refers to the number of query documents with rank equal to 1 and TOP-5 refers to the number of a query with rank equal and smaller than 5.

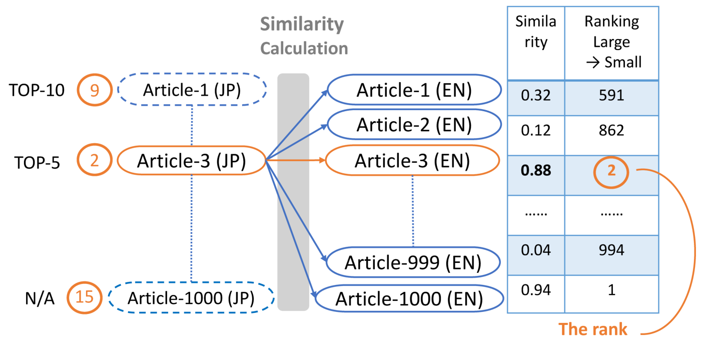


Figure 3. Illustration of an evaluation procedures using ranks and TOP-N index

4.2. Baseline: Siamese LSTM with Google-translation

As Siamese LSTM is one of the deep learning-based models with the art-of-state performance on the semantic text similarity problems. In this research, we make this model be a baseline by extending this model from monolingual domain to cross-lingual domain with the help of the Google Translation services. We first translate all Japanese text into English version on both test and training data by using the google translate service [1] . Then we implement the Siamese LSTM model as described in the original paper for Siamese LSTM [10] with the help of the open source code on the Github [2]. The illustration of this baseline method regarding a two cross-lingual input, we first translate the Japanese input into an English sentence using Google Translation service. Then, we can consider the cross-lingual task as monolingual one using so that we can apply the Siamese LSTM model for training as a baseline.

4.3. Datasets and Pre-processing

Thomson Reuters news [3] is a worldwide news agency providing worldwide news in multiple languages. Most of the reports are originally written in English and translated and edited into other languages including Chinese, Japanese and so on. These multi-lingual texts are expected to be highly potential resources for tasks related the multi-lingual natural languages processing. In this research, we use 60,000 news articles in 2014 from Thomson Reuters News related to the economics. The

[1] Google Translation Web API could be accessed from https://github.com/aditya1503/Siamese-LSTM
[2] The open source code for Siamese LSTM can be accessed from https://github.com/aditya1503/Siamese-LSTM
[3] Official websites of Thomson Reuters: http://www.reuters.com/

Table 1. Example of similarity relationship for Japanese words (translated)

	Toyota		Sony	
TOP	word	Similarity	word	Similarity
1	Honda	0.612	PlayStation	0.612
2	Toyota corp	0.546	Entertainment	0.546
3	Hyundai corp	0.536	SonyBigChance	0.536
4	Chrysler	0.524	Game console	0.524
5	Nissan	0.519	Nexus	0.519
6	motor	0.511	X-BOX	0.511
7	LEXUS	0.506	spring	0.506
8	Acura	0.493	Windows	0.493
9	Mazda	0.492	Compatibility	0.492
10	Ford	0.486	application software	0.486

preprocessing of text, we convert raw data to normalized ones, which could be further used to train word2vec models for both English and Japanese text respectively. We train Japanese word2vec model and English word2vec model separately using news articles with same contents in 2014. In our experiment, we use the model of Continuous Bag of Words (CBOW), with fixed 200 dimensions of word embedding. Other parameters are set using default value used in Gensim package [4].

Table 2. Example of similarity relationship for English words

	lexus		lenovo	
TOP	word	Similarity	word	Similarity
1	acura	0.636	huawei	0.636
2	corolla	0.588	zte	0.588
3	camry	0.571	xiaomi	0.571
4	2002-2005	0.570	dell	0.570
5	sentra	0.541	handset	0.541
6	prius	0.539	smartphone	0.539
7	2003-2005	0.537	hannstar	0.537
8	sedan	0.533	thinkpad	0.533
9	mazda	0.530	tcl	0.530
10	altima	0.524	medison	0.524

As discussed in the section 3.1, the word2vec could build relationships among words based on their original context. We could find several most similar words when given a query word by calculating their cosine similarity. The table 2 and 1 demonstrate examples to find the most similar words when given a word query in English and in Japanese respectively. All these results suggest the effectiveness of word2vec algorithms and successful of the training processes.

4.4. Experiments on Text Datasets

4.4.1. Training Data

On Short Text, we firstly pick up 4000 pairs of parallel Japanese-English cross-lingual news titles from the database with the period from the January to February of 2014, all of which are labeled with a similarity score of 1. To provide balance training data, we also generated 4000 pairs of un-parallel

[4] To see more specific of the configuration of word2vec model, see the documentation of Word2Vec class from https://radimrehurek.com/gensim/models/word2vec.html

Japanese-English cross-lingual news titles by a random combination. In order to simplify our model and experiments, we use the assumption that the similarity the random combination of Japanese text and English text is 0. Then on Long Text, similar to the data preparation of experiments for short text introduced, we prepare 4000 parallel (i.e. similarity = 1) Japanese-English news articles and 4000 un-parallel (i.e. similarity = 0) ones for training data through random combination.

4.4.2. Test Data

On Short Text, in order to evaluate our model more comprehensively, we have prepared two sets of independent test data. TEST-1S contains 1000 pairs of parallel Japanese-English news titles, selected and split from the same period of training data, from January 2014 to middle of February in 2014. Similar, TEST-2S contains title pairs with time stamps of December 2014. On Long Text, similar to the case of short test evaluation, we have prepared two sets of independent test data. For training data, we prepared similar dataset as short text experiments. TEST-1L and TEST-2L contain 1000 pairs of parallel Japanese-English news long articles respectively.

4.4.3. Ranks and TOP-N

Table 3 also summarizes and compares our two purposed models, LSTM-based model and SVM-based model respectively in terms of TOP-N benchmark regarding LONG text scenario and SHORT text scenario.

First, we could notice that both our purposed, LSTM-based model and SVM-based model, outperform the baseline in terms of all three TOP-N criteria. In terms of TOP-N, the LSTM-based model obtains around twice the performance of the baseline (511 vs. 243) on short test data, while LSTM-based model also has twice the performance of the baseline (685 vs. 302) on long test data, suggesting the effectiveness and efficiency of our purposed models.Furthermore, we may also easily notice that SVM-based method outperforms the LSTM-based methods in terms of TOP-N criteria, in case of long text, around 50%. In contrast, the LSTM-based model has a TOP-10 score around 10% higher than that of the SVM-based model on both two test datasets.

The dominant performance of the SVM-based model on long test data maintains also in terms of TOP-1 and TOP-5, twice the score compared to LSTM-based model for TOP-5 and three times the score for the TOP-1 benchmark. On the other hand, although the LSTM-based model still performs better than SVM-based with respect to TOP-5, as for TOP-5 LSTM-based model failed to be in the lead anymore. We are going to discuss these results and purpose possible hypothesis and provide explanations in the section 5. The performance of successful recommendation numbers from our bi-LSTM based model is twice of the baseline.

5. Discussion

5.1. Comparison of the baseline and the LSTM-based model

The performance of the LSTM-based model is twice of the baseline, even though they are both based on LSTM structures. The differences, which are also the innovations for this purposed method, compared to the baseline, include the using of bi-LSTM, independent LSTM modules as well as using the fully connected neural network as the final layer.

First, the baseline method could calculate the similarity of two sentences, no matter whether there are different types of word arrangement for the two input, or there are different words used referring to the same meaning, which proves the effectiveness of encoding (i.e. embedding) ability for input text. However, the baseline model has the “Siamese LSTM structure", which means, in other words, the two LSTM instances always share the same parameters during the training. This might be effective for a monolingual case, but not good enough on the cross-lingual case. Thus, the LSTM instances used in our purposed model are all independently holding their own unique parameters. In addition, the bi-directional structure also helps to encode the feature of each input text more comprehensively.

Table 3. Summary of in terms of TOP-N benchmark

TOP-10				
	SHORT		LONG	
	TEST-1S	TEST-2S	TEST-1L	TEST-1L
LSTM	**511**	**495**	456	432
SVM	453	422	**685**	**654**
baseline	243	-	302	-

TOP-5				
	SHORT		LONG	
	TEST-1S	TEST-2S	TEST-1L	TEST-1L
LSTM	**339**	**338**	284	278
SVM	324	295	**520**	**491**
baseline	134	-	192	-

TOP-1				
	SHORT		LONG	
	TEST-1S	TEST-2S	TEST-1L	TEST-1L
LSTM	90	**106**	61	58
SVM	**101**	96	**128**	**179**
baseline	39	-	50	-

Finally, instead of using cosine similarity as the final layer in the baseline method, we used the fully connected neural network as output, making the output layer adjust (i.e. train) its parameters so as to learn precise patterns from the features generated by LSTMs. We believe these three modifications improve the final results for our LSTM-based model.

5.2. Comparison of the LSTM-based model and SVM-based model

The experiments above leave us an interesting question about why LSTM-based model and SVM-base model performs differently regarding to the length of the target text we train and test. We explain this question in two aspects.

5.2.1. From the point of view of the SVM-based model

Since the SVM-based methods use the TF-IDF weighting which is a classical and an effective method for NLP fields to extract the most important and representative features for each of document comprehensively, it could accurately identify the most significant feature, a few key words, from a very long and complex article containing hundreds of words, in both Japanese and English, and then finally feed them into SVM classifier to get the similarity estimation universally. However, due to the attributes of TF-IDF algorithms, shorten the length of each document is, less information could the TF-IDF extract. This is because if there is only a fewer words in one document, every word could be either unique or common regarding to other documents, resulting in the failure of TF-IDF. This might be the reason why SVM-based model performs well on long datasets but becomes poor on shorter data sets.

5.2.2. From the point of view of the LSTM-based model

On the other hand, the LSTM is good at understand sentences by means of grasping the order information of each words, since for any natural languages, not only words themselves but also the order of words, to some extends, define the true meaning of a sentence. Especially as for short text, the slight change of the order could alter the meaning of the sentences significantly and thus LSTM-based model outperformance the LSTM model around 10% on short datasets. However, LSTM is not good at extracting the key idea of longer documents since although LSTM solves the problem of memorizing long text (i.e. solve of the problem of gradient vanishing and gradient explosion), it could tell the importance of each word as TF-IDF does. That might be the possible reason why it fails to perform effectively on a long text.

6. Conclusion

We developed a bi-LSTM-based model to calculate cross-lingual similarities given a pair of English and Japanese articles. Instead of using a translation module or a dictionary to translate from one to another language, our model has outstanding performance with short text. Furthermore, we modify and implement a popular Siamese LSTM model as the baseline and we found both of our models outperform the baseline. For practical testing, we defined the concept of "TOP-N" and "ranks" to test the overall performance of the model, with visualized results. We also make a comparative study based on the results of the experiments that bi-LSTM based obtains better performance on short text data such as news title and alert message, which is averagely shorter than 20 words, a contrast to the normal news articles with more than 200 words in average. As the results, both models obtained satisfactory performance with over half of the test documents of 1000 holding ranks lower than 10 (i.e. TOP-10). As a high-performance cross-lingual news calculating system, we expect that it could achieve optimal performance by taking advantages of both two models as a complete system.

References

1. Tang, D.; Wei, F.; Yang, N.; Zhou, M.; Liu, T.; Qin, B. Learning Sentiment-Specific Word Embedding for Twitter Sentiment Classification. ACL (1), 2014, pp. 1555–1565.
2. Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.S.; Dean, J. Distributed representations of words and phrases and their compositionality. Advances in neural information processing systems, 2013, pp. 3111–3119.
3. Le, Q.; Mikolov, T. Distributed representations of sentences and documents. Proceedings of the 31st International Conference on Machine Learning (ICML-14), 2014, pp. 1188–1196.
4. Zou, W.Y.; Socher, R.; Cer, D.; Manning, C.D. Bilingual word embeddings for phrase-based machine translation. Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, 2013, pp. 1393–1398.
5. Gouws, S.; Bengio, Y.; Corrado, G. Bilbowa: Fast bilingual distributed representations without word alignments. Proceedings of the 32nd International Conference on Machine Learning (ICML-15), 2015, pp. 748–756.
6. Rupnik, J.; Muhic, A.; Leban, G.; Skraba, P.; Fortuna, B.; Grobelnik, M. News across languages-cross-lingual document similarity and event tracking. *Journal of Artificial Intelligence Research* **2016**, *55*, 283–316.
7. Kudo, T.; Yamamoto, K.; Matsumoto, Y. Applying Conditional Random Fields to Japanese Morphological Analysis. EMNLP, 2004, Vol. 4, pp. 230–237.
8. Taghva, K.; Elkhoury, R.; Coombs, J. Arabic stemming without a root dictionary. Information Technology: Coding and Computing, 2005. ITCC 2005. International Conference on. IEEE, 2005, Vol. 1, pp. 152–157.
9. Lo, C.k.; Beloucif, M.; Saers, M.; Wu, D. XMEANT: Better semantic MT evaluation without reference translations. Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), 2014, Vol. 2, pp. 765–771.
10. Mueller, J.; Thyagarajan, A. Siamese Recurrent Architectures for Learning Sentence Similarity. AAAI, 2016, pp. 2786–2792.
11. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural computation* **1997**, *9*, 1735–1780.
12. Agirre, E.; Banea, C.; Cer, D.M.; Diab, M.T.; Gonzalez-Agirre, A.; Mihalcea, R.; Rigau, G.; Wiebe, J. SemEval-2016 Task 1: Semantic Textual Similarity, Monolingual and Cross-Lingual Evaluation. SemEval@ NAACL-HLT, 2016, pp. 497–511.
13. Baroni, M.; Dinu, G.; Kruszewski, G. Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. ACL (1), 2014, pp. 238–247.
14. Burges, C.J. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery* **1998**, *2*, 121–167.

15. Malakasiotis, P.; Androutsopoulos, I. Learning textual entailment using SVMs and string similarity measures. Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing. Association for Computational Linguistics, 2007, pp. 42–47.
16. Béchara, H.; Costa, H.; Taslimipoor, S.; Gupta, R.; Orasan, C.; Pastor, G.C.; Mitkov, R. MiniExperts: An SVM Approach for Measuring Semantic Textual Similarity. SemEval@ NAACL-HLT, 2015, pp. 96–101.
17. Schroff, F.; Kalenichenko, D.; Philbin, J. Facenet: A unified embedding for face recognition and clustering. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 815–823.
18. Graves, A.; others. *Supervised sequence labelling with recurrent neural networks*; Vol. 385, Springer, 2012.
19. Greff, K.; Srivastava, R.K.; Koutník, J.; Steunebrink, B.R.; Schmidhuber, J. LSTM: A search space odyssey. *IEEE transactions on neural networks and learning systems* **2017**.
20. Nair, V.; Hinton, G.E. Rectified linear units improve restricted boltzmann machines. Proceedings of the 27th international conference on machine learning (ICML-10), 2010, pp. 807–814.
21. Kingma, D.; Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* **2014**.