electronics

MDPI

Article

Translating Sentimental Statements Using Deep Learning Techniques

Yin-Fu Huang * and Yi-Hao Li

Department of Computer Science and Information Engineering, National Yunlin University of Science and Technology, Yunlin 64002, Taiwan; jack1995319@gmail.com

* Correspondence: huangyf@yuntech.edu.tw; Tel.: +886-5534-2601 (ext. 4314)

Abstract: Natural Language Processing (NLP) allows machines to know nature languages and helps us do tasks, such as retrieving information, answering questions, text summarization, categorizing text, and machine translation. To our understanding, no NLP was used to translate statements from negative sentiment to positive sentiment with resembling semantics, although human communication needs. The developments of translating sentimental statements using deep learning techniques are proposed in this paper. First, for a sentiment translation model, we create negative–positive sentimental statement datasets. Then using deep learning techniques, the sentiment translation model is developed. Perplexity, bilingual evaluation understudy, and human evaluations are used in the experiments to test the model, and the results are satisfactory. Finally, if the trained datasets can be constructed as planned, we believe the techniques used in translating sentimental statements are possible, and more sophisticated models can be developed.

Keywords: deep learning; natural language processing; text mining; semantics

Citation: Huang, Y.-F.; Li, Y.-H. Translating Sentimental Statements Using Deep Learning Techniques. *Electronics* **2021**, *10*, 138. https://doi.org/10.3390/electronics10020138

Received: 5 December 2020
Accepted: 7 January 2021
Published: 10 January 2021

1. Introduction

To date, Natural Language Processing (NLP) methods have been commonly applied in many applications. NLP allows machines to know nature languages and helps us do tasks, such as retrieving information [1], answering questions [2], text summarization [3], categorizing text [4], and machine translation [5,6].

Recently, current Seq2seq methods such as Seq2seq [6], SeqGAN [7], and transformer [8] have become effective in text generation. A multilayered Long Short-Term Memory (LSTM) [6] was used to map the input sequence to a fixed dimensionality vector. Then, another deep LSTM was used to decode the target sequence from the vector. The application is for an English to French translation task. SeqGAN [7] proposed a sequence generation framework to solve the gradient update problems from the discriminative model to the generative model. Modeling the data generator as a stochastic policy in reinforcement learning (RL), SeqGAN bypasses the generator differentiation problem by directly performing gradient policy updates. The RL reward signal comes from the GAN discriminator judged on a complete sequence SeqGAN [7] proposed a sequence generation framework to solve and is passed back to the intermediate state-action steps using the Monte Carlo search. A transformer is a new deep learning model introduced in Dec. 2017 [8] and used primarily in natural language processing. A transformer is designed to handle sequential data, but it does not require that the sequential data are processed in the order. Since the transformer model facilitates more parallelization during training, it has enabled training on larger datasets.

To our understanding, no NLP has been used to translate statements from negative sentiment to positive sentiment with resembling semantics, despite the human communication needs. The Positive Sentimental Statement (PSS) "John is always late, and we should be concerned with him" can be translated from the Negative Sentimental Statement (NSS) "John is always late, which makes us annoyed". The PSS can be conveyed in this translation from

pleasant viewpoints. Generally, various cultures have their own customs and communication habits, so rude communication may occur when people do not understand each other.

The developments of translating sentimental statements using deep learning techniques are proposed in this paper. The Sentiment Translation Model (STM) is assumed to be trained by using negative–positive pairs of sentimental statements with resembling semantics to construct an STM for translating statements from negative sentiment to positive sentiment. First, via web crawlers, almost 110 million Amazon Product Reviews and Ratings (APRR) are obtained from Amazon. To train the STM, the Negative–Positive Sentimental Statement (NPSS) datasets are then constructed with over 6 million negative–positive pairs of sentimental statements with resembling semantics. Here for deep learning, the STM uses the LSTM-based Seq2seq model. Finally, in applications such as chat messages, the trained STM can translate statements from negative sentiment to positive sentiment. In summary, in this paper, we characterize the novelty and contributions below:

1. The definitions of negative–positive pairs of sentimental statements with resembling semantics are proposed.
2. To train the STM, the novel NPSS datasets are constructed with over 6 million negative–positive pairs of sentimental statements with resembling semantics.
3. A series of tests are carried out to determine the translation outcomes, and further human evaluations on these outcomes are presented.

The paper is organized as follows. The system design for translating statements from negative sentiment to positive sentiment is illustrated in Section 2. In Section 3, we create negative–positive pairs of sentimental statements with resembling semantics for training the STM. In the STM for deep learning, the LSTM-based Seq2seq model with the attention mechanism is presented in Section 4. In Section 5, to test the STM, perplexity, bilingual evaluation understudy, and human evaluations are used in the experiments. Finally, in Section 6, we make conclusions and propose future work.

2. System Overview

In this section, as shown in Figure 1, we briefly explain how to construct an STM to translate statements from negative sentiment to positive sentiment. The STM is presumed to be trained by using negative–positive pairs of sentimental statements with resembling semantics. The PSS might be "John is always late, and we should be concerned with him". against the NSS "John is always late, which makes us annoyed".

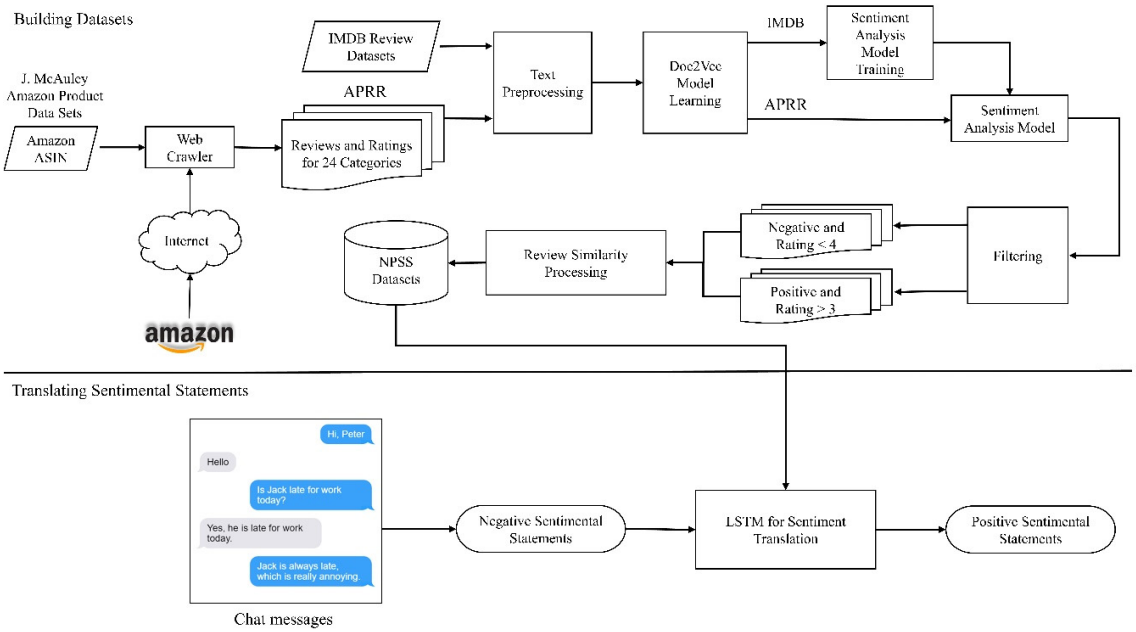

Figure 1. System architecture.

The Amazon Standard Identification Number (ASIN) is obtained from the J. McAuley Amazon product datasets [9,10] in the building dataset phase. ASIN is a special identification number composed of ten characters that can be used on the Amazon website to identify items using letters or numbers. With 24 product categories in the J. McAuley Amazon product datasets, a web crawler is used to grab APRR from the Amazon website. Furthermore, we also collect datasets for the IMDB review [11] consisting of movie reviews with labels. Then, both IMDB and APRR datasets go through text preprocessing and enter into the learning Doc2Vec model [12], which transforms a document into a vector. Next, in the Sentiment Analysis (SA) model, logistic regression is trained by using IMDB reviews, and the SA model tags positive or negative views on APRR. As 24 product categories are processed separately, 24 Doc2Vec models and SA models are generated during training. A rule must then be used to filter the APRR tagged by the SA models, i.e., either the product rating of a negative-tagged review must be less than 4, or the product rating of a positive-tagged review must be more than 3. Here we retain the consistent product reviews, and the cosine similarity is used with each negative review to find the highest-similarity positive review. Finally, the NPSS datasets are used to train the STM, where negative–positive pairs of reviews are gathered.

If an NSS occurs in the chat messages in the translating sentimental statement phase, the learned STM will translate the statement from negative sentiment to positive sentiment. Here for deep learning, the STM utilizes the LSTM-based [13] Seq2seq model [6].

3. Building Datasets

Although many negative–positive pairs of reviews are available on the WWW, they have only been used in the sentiment analysis. They have never been extended to sentiment translation applications yet. It could be that either negative–positive pairs of sentimental statements did not co-exist with resembling semantics, or no one has ever investigated their opposite sentimental statements from sentimental statements with resembling semantics. We generated negative–positive pairs of sentimental statements with resembling semantics in the building dataset phase for later training of the STM.

The proposed method's detailed process in the building dataset phase is outlined as follows. In Section 3.1, both IMDB and APRR datasets used in the study are introduced first. The texts in the IMDB and APRR reviews are then preprocessed using the language recognition tool langid.py in Section 3.2. Thus, the noises affecting the model learning will be filtered out from the IMDB and APRR reviews. Next, in Section 3.3, clean IMDB and APRR datasets enter into the Doc2Vec model using the Python Gensim library for learning semantics, which eventually quantifies a document into an N-dimensional vector. Then, in Section 3.4, the logistic regression of the scikit-learn library is used to train the SA model using the IMDB reviews, and the SA model can tag positive or negative labels on APRR. Next, to keep product reviews consistent with the Amazon product rating, the APRR reviews tagged by the SA model are filtered in Section 3.5. Next, for each negative review, the cosine similarity can be used to find the highest-similarity positive review in Section 3.6. Finally, negative–positive pairs of reviews collected in the NPSS datasets are used to train the STM.

3.1. Datasets

J. McAuley Amazon product datasets [9,10] provide product metadata on the Amazon website, such as reviews, ratings, ASIN, and more. According to ASIN, we run web crawlers to capture complete reviews of products on the Amazon website. In Python 3.6.0, using the Selenium and Beautiful Soup libraries, the web crawler is implemented. Selenium is a graphical web tool that enables users to easily log browser actions and complete partial tests quickly. Moreover, Beautiful Soup is used to process data in HTML and XML files. The web crawler served from 28 February 2018 to 25 May 2018. In total, as shown in Table 1, 24 categories of Amazon products with 1.1 million items and almost 110 million APRR were obtained. Among them, approximately one-third of items are occupied by the category book, and the ratio of 4~5 to 1~3 ratings is 4:1.

Table 1. Twenty-four categories of Amazon products.

	Categories	No. of Products	Ratings 4~5	Ratings 1~3	No. of Reviews
C1	Apps for Android	13,147	2,006,327	778,726	2,785,053
C2	Automotive	1817	709,782	145,202	854,984
C3	Baby	6759	1,276,168	355,542	1,631,710
C4	Beauty	10,825	2,533,868	712,946	3,246,814
C5	Book	365,156	25,550,896	5,001,205	30,552,101
C6	CDs and Vinyl	64,443	3,202,221	495,215	3,697,436
C7	Cell Phones and Accessories	10,396	2,038,401	869,234	2,907,635
C8	Clothing Shoes and Jewelry	22,787	4,473,465	1,172,548	5,646,013
C9	Electronics	62,857	8,698,745	2,683,703	11,382,448
C10	Grocery and Gourmet Food	8560	1,962,621	386,246	2,348,867
C11	Health and Personal Care	18,458	4,924,158	1,240,566	6,164,724
C12	Home and Kitchen	28,222	6,561,360	1,756,013	8,317,373
C13	Kindle Store	61,929	2,788,628	647,101	3,435,729
C14	Movies and TV	50,576	6,487,984	1,425,056	7,913,040
C15	Musical Instruments	898	334,020	55,339	389,359
C16	Office Products	2329	1,045,263	255,911	1,301,174
C17	Patio Lawn Garden	954	464,412	122,039	586,451
C18	Pet Supplies	8510	2,763,138	792,482	3,555,620
C19	Reviews Amazon Instant Video	20,425	943,180	199,980	1,143,160
C20	Reviews Digital Music	263,648	1,600,171	151,627	1,751,798
C21	Sports and Outdoors	18,296	3,590,097	750,991	4,341,088
C22	Tools and Home Improvement	10,097	2,261,018	512,217	2,773,235
C23	Toys and Games	11,906	1,962,085	435,060	2,397,145
C24	Video Games	8160	759,230	237,912	997,142
All		1,071,155	88,937,238	21,182,861	110,120,099

The IMDB dataset consisting of positive and negative reviews is another one used in the paper. IMDB has 100,000 reviews where a half with label reviews are for supervised learning, and another half with non-label reviews are for unsupervised learning. There are at most 30 reviews of each movie.

3.2. Text Preprocessing

Before training the Doc2Vec and SA models, the texts in the IMDB and APRR reviews are preprocessed; i.e., the noises that affect the learning model have to be removed from the reviews. All the punctuation marks other than dots, exclamation marks, commas, and question marks were canceled. Moreover, we removed emoticons such as :-), :c, :-x), replaced several consecutive spaces with a single space, and deleted the reviews' URL. We also deleted the non-English reviews using the language recognition tool called langid.py.

3.3. Doc2Vec Model Learning

A very significant step in NLP is the translation of texts into vectors. Sentiment analysis applications generally use logistical regression, support vector machine, convolutional neural network, and recurrent neural network. However, normally these algorithms require a feature vector with a fixed length. We first implement the Doc2Vec model in this section and then specify the dimensions and training algorithms used in the model. Here, both the IMDB and APRR datasets after text preprocessing enter into the Doc2Vec model.

3.3.1. Doc2Vec Model

The bag-of-words model, the bag of n-grams model, and the average word vector are common fixed-length text vector representation techniques. The bag-of-words model considers the terms in sentences or files and the occurrence frequency. It produces a vocabulary of all the specific words that appear in the texts, such as [Rose, likes, to, watch, TV, John, play, baseball], for two text documents, "Rose likes to watch TV", and "John likes to play baseball". Then to record the frequencies of all the different terms, it builds

two lists such as [1, 1, 1, 1, 1, 0, 0, 0] and [0, 1, 1, 0, 0, 1, 1, 1, 1]. The bag-of-words model does not, however, take into account the order between words, i.e., as long as they have the same words, different sentences have the same representation. The bag of n-grams model is a natural bag-of-words extension that treats n-gram words as a vocabulary and counts their incidence frequencies. 2-gram's unique list of words for the above example is [Rose likes, likes to, to watch, watch TV, John likes, likes to, to play, play baseball]. The two frequency lists built by the 2-gram's unique list of words are [1, 1, 1, 1, 0, 0, 0, 0] and [0, 0, 0, 0, 1, 1, 1, 1]. However, the bag of n-grams model requires a longer vector and causes a problem of high dimensionality and sparsity. Finally, the word vector, which is the beginning of the word distribution representation, is a distributed representation of concepts proposed by Hinton [14]. Word vectors have a direct definition of word-to-word similarity, and two efficient word vector algorithms word2vec [15–17] and GloVe [18], are then generalized. All the word vectors in the sentence were averaged by Hong and Fang [4] to get the sentence vector, but this still has the drawback of losing the detailed word order. Subsequently, Le and Mikolov [12] proposed paragraph vectors, which perform well in text vectors. The definition of its training is similar to the word2vec algorithm.

Word2vec was proposed by Google's team [15–17], which can be a simple neural network model with two training algorithms, i.e., continuous bag of words (CBOW) and Skip-Gram. The CBOW architecture is based on a feedforward neural network language model (NNLM) proposed by Bengio et al. [19]. Both algorithms need to learn word vectors from large-scale corpora. After the model is trained, words with similar meanings are mapped to similar vector space positions. Obtaining the dimensional vector representation of each word makes it easy to calculate the semantic similarity between words. It can also use the word vector to calculate the meaning between words; for example, the result of vector(King) minus vector(Man) plus vector(Woman) would be close to vector(Queen).

The CBOW architecture consists of input, projection, and output layers, which uses contexts to predict the current target word. The input of the input layer is the word vector of the words surrounding the current target word. Each word vector uses a one-hot encoding representation, which is a long vector. The length of the vector depends on the size of the word vocabulary. Only one dimension is 1, and all others are 0. The position of 1 corresponds to the position of the word in the dictionary. The weight matrix between the input and projection layers is shared for all word positions in the same way as in NNLM. The output layer is a classifier to maximize the probability of the current target word, as shown in Figure 2.

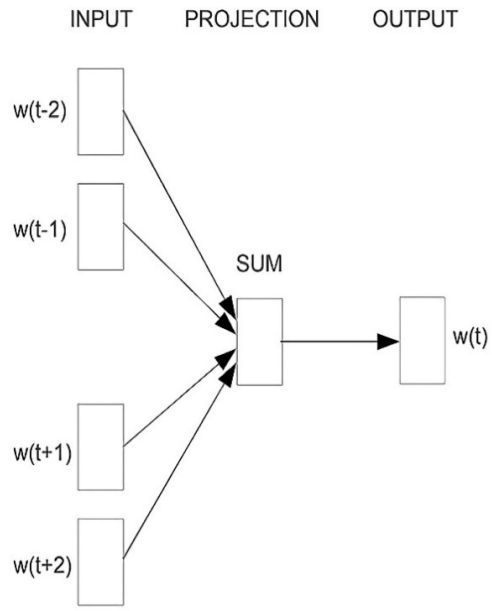

Figure 2. Current target word predicted by continuous bag of words (CBOW).

The Skip-Gram architecture is similar to CBOW and consists of input, projection, and output layers. However, Skip-Gram uses a word to predict the contexts; in other words, the input layer's input is only the current word, and the target is the surrounding words of the current word, as shown in Figure 3.

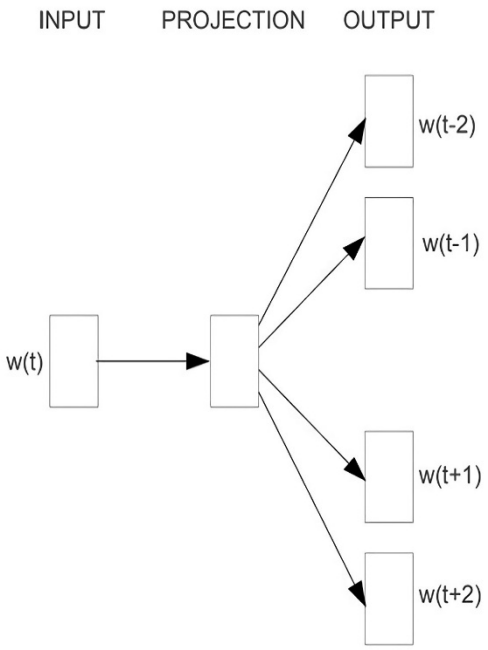

Figure 3. Surrounding words predicted by Skip-Gram.

Doc2Vec is also known as Paragraph vector, proposed by Le and Mikolov [12], who worked for Google in 2014. Paragraph vector technology was inspired by other work using neural networks to learn word vector representations. It is an unsupervised framework for learning continuous distributed vector representations of text fragments. The learned vectors can be used to find the similarity between texts by calculating their distances. Moreover, their word vector representation can also be used in supervised learning methods for sentiment analysis applications for labeled data. This method can be applied to variable-length texts, including phrases, sentences, and even documents. The paragraph vector is an extension of Word2vec, and the concept is similar to the Word2vec algorithm. There are two versions of the training algorithm: (1) distributed memory model of paragraph vectors (PV-DM) and (2) distributed bag of words model of paragraph vectors (PV-DBOW).

PV-DM is similar to Word2vec's CBOW architecture. The only difference is that it adds a feature vector to represent a paragraph id. Each paragraph is mapped to a unique vector, which can be represented by a column of matrix D, and each word is also mapped to a unique vector, which can be represented by a column of matrix W. The paragraph and word vectors are averaged or concatenated to predict the current word, and its paragraph id remains the same when training the same paragraph, as shown in Figure 4.

PV-DBOW is similar to Word2vec's Skip-Gram architecture. It ignores input context and adds a vector representing a paragraph id to the model to predict the paragraph's words, as shown in Figure 5.

In the original paper, Quoc et al. recommended using a combination model of PV-DM and PV-DBOW, although the PV-DM model alone usually works well for most tasks.

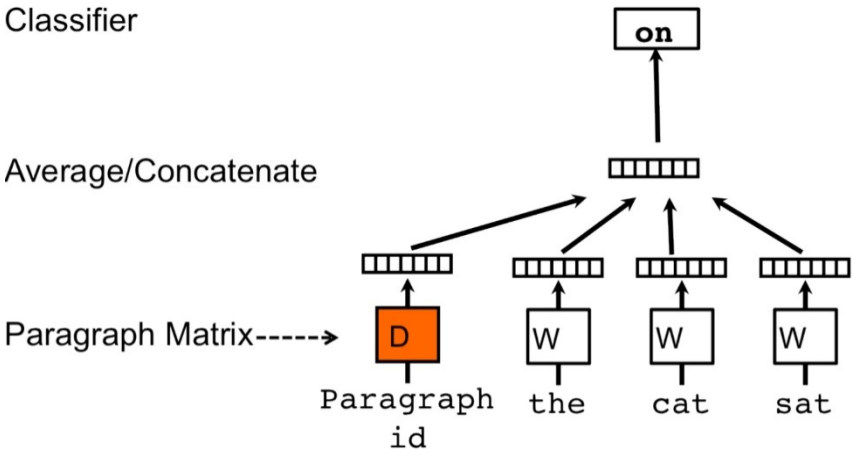

Figure 4. Distributed memory model of paragraph vectors (PV-DM) architecture.

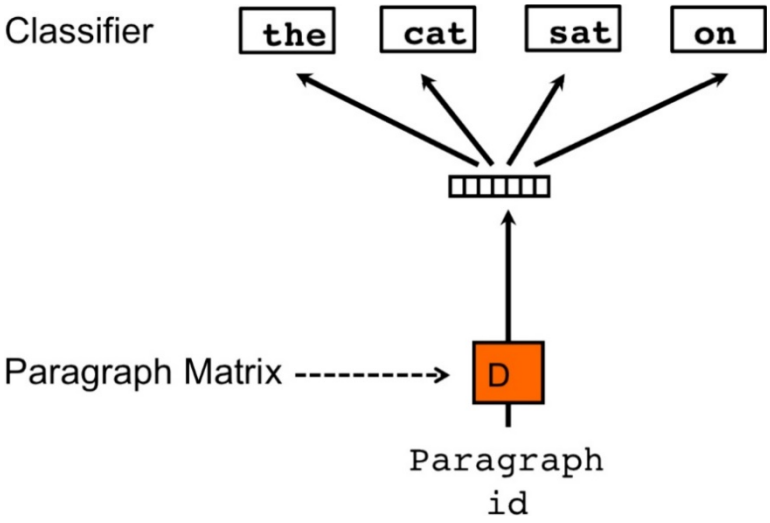

Figure 5. Distributed bag of words model of paragraph vectors (PV-DBOW) architecture.

3.3.2. Determining Dimensions and Architectures

The Doc2Vec model is essential in generating feature vectors for sentiment classification and finding similar semantic sentences. Here, dimensions and architectures for the Doc2Vec model learning must be considered and determined. We test 100, 150, 250, and 400 dimensions to achieve the best performances. For the architectures, we test several models: (1) distributed bag of words (DBOW), (2) distributed memory means (DMM), (3) distributed memory concatenation (DMC), and (4) all combinations. Moreover, we also consider four combinations in review contents: (1) no punctuations and stop-words (NPS), (2) no stop-words (NS), (3) no punctuations (NP), and (4) keeping punctuations and stop-words (PS). In practice, we use the Python Gensim library [20] to implement the Doc2Vec model, which was specifically designed to handle large texts and has been implemented in text processing related algorithms such as TF-IDF, random projections, Word2vec, Doc2vec, Latent Dirichlet Allocation (LDA), and more. [21]. Owing to labeled data in the IMDB dataset, we use them to generate the feature vectors and then use the SA model to evaluate the quality of the Doc2Vec model. The training goes through 25 epochs, and all the parameters are default, except window size 8 for DMM and window size 5 for DMC. The best evaluation results (or accuracy %) for dimensions and architectures are shown in Table 2. (Bold indicate the best accuracy among all architectures).

Table 2. Best evaluation results (or accuracy %) for dimensions and architectures.

Dimensions	Architectures	NPS	NS	NP	PS
100	DBOW	89.32	89.58	89.56	89.58
	DMM	87.32	87.60	87.13	86.78
	DMC	82.85	81.64	83.36	83.56
	DBOW + DMM	**89.54**	**89.56**	**89.67**	**89.68**
	DBOW + DMC	89.35	89.55	89.61	89.56
	DMM + DMC	86.58	86.60	86.48	86.24
150	DBOW	89.20	89.15	89.36	89.26
	DMM	87.82	87.89	87.73	87.64
	DMC	81.80	82.54	83.03	83.26
	DBOW + DMM	**89.40**	**89.56**	**89.76** *	**89.60**
	DBOW + DMC	89.24	89.27	89.45	89.38
	DMM + DMC	86.16	86.60	86.18	86.31
250	DBOW	89.21	89.10	89.02	88.95
	DMM	87.40	87.60	87.64	87.60
	DMC	80.95	82.04	82.56	82.77
	DBOW + DMM	**89.39**	**89.32**	**89.26**	**89.35**
	DBOW + DMC	89.18	89.12	89.02	88.98
	DMM + DMC	85.99	85.85	86.11	86.06
400	DBOW	89.15	89.13	88.89	88.95
	DMM	87.74	87.81	87.95	87.96
	DMC	76.88	80.49	73.61	73.65
	DBOW + DMM	**89.27**	**89.17**	**89.13**	**89.01**
	DBOW + DMC	89.21	89.16	89.05	88.94
	DMM + DMC	85.48	85.68	85.38	85.31

* indicate the selected model is DBOW + DMM for no punctuations (NP).

We think that determining architectures is more important than determining dimensions and review contents. The combination of DBOW and DMM always achieves an accuracy higher than 89% in all the test dimension cases. According to the best accuracy in the evaluation results, the combination of DBOW and DMM (with dimensions 150 and no punctuations) is determined and used in the Doc2Vec model learning. Finally, both IMDB and APRR for each category regarded as a dataset enter into the confirmed Doc2Vec model to generate feature vectors for learning semantics. Since products with 24 categories are processed separately, we produce 24 Doc2Vec models.

3.4. Sentiment Analysis Model

To recognize subjective information in texts, sentiment analysis was used, also known as opinion mining [22]. Here the goal of the sentiment analysis is to tag APRR with positive or negative labels. In the SA model, logistic regression is trained and tested using IMDB reviews. For 50,000 IMDB label reviews used in the supervised learning, 25,000 reviews are for training and another 25,000 reviews are for the test, and both have the same number of positive and negative reviews. Because 24 Doc2Vec models are used separately to generate feature vectors, 24 SA models are trained and tested, and Table 3 shows their best accuracy %. Finally, these 24 SA models can tag APRR with positive or negative labels.

Table 3. Best accuracy % of 24 sentiment analysis (SA) models.

Category	C1	C2	C3	C4	C5	C6	C7	C8
Accuracy	88.74	89.19	88.58	88.12	87.63	87.96	88.17	87.85
Category	C9	C10	C11	C12	C13	C14	C15	C16
Accuracy	86.48	88.32	87.73	87.20	88.12	88.69	89.24	88.73
Category	C17	C18	C19	C20	C21	C22	C23	C24
Accuracy	89.14	87.92	89.08	88.70	87.82	88.12	88.16	89.10

3.5. Filtering

A rule must be used to filter the APRR which are tagged by the SA models, i.e., either the product rating of a negative-tagged review must be less than 4, or the product rating of a positive-tagged review must be more than 3. Here we retain the consistent product reviews. Generally, product reviews of grades 4 and 5 are considered positive, and product reviews of grades 1 and 2 are considered negative. Since negative reviews are far smaller than positive reviews, as seen in Table 1, we treat neutral reviews (with rating 3) as negative reviews. Moreover, using boxplots as shown in Figure 6, we calculate the dispersion of review lengths in 24 categories.

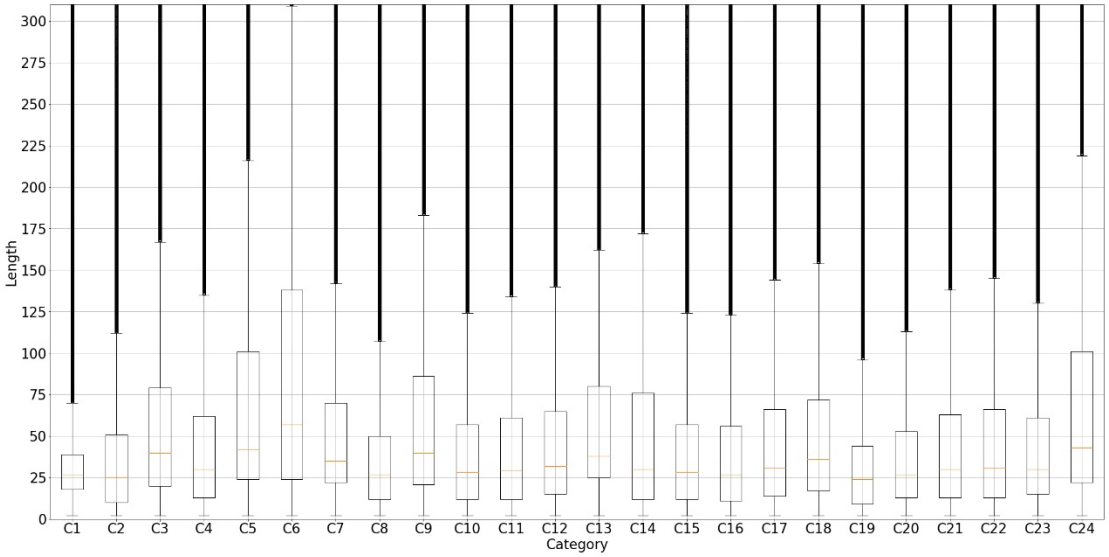

Figure 6. Boxplots of review lengths (no. of tokens) in 24 categories.

3.6. Review Similarity Processing

The cosine similarity is used to find the highest-similarity positive review of a negative one from the same category. Cosine similarity is a similarity-calculating method used in information retrieval. The similarity between two vectors can be calculated, and the similarity ranges in $[-1, 1]$. Value -1 indicates that two vectors are completely unlike, 1 indicates the same, and 0 indicates independent. The formula for cosine similarity is as follows:

$$\text{similarity}(Neg,\ Pos) = \text{DBOW}\left(\frac{\sum_{i=1}^{n} Neg_i \times Pos_i}{\sqrt{\sum_{i=1}^{n}(Neg_i)^2} \times \sqrt{\sum_{i=1}^{n}(Pos_i)^2}}\right) + \text{DMM}\left(\frac{\sum_{i=1}^{n} Neg_i \times Pos_i}{\sqrt{\sum_{i=1}^{n}(Neg_i)^2} \times \sqrt{\sum_{i=1}^{n}(Pos_i)^2}}\right) \quad (1)$$

where *Neg* is the feature vector of a negative review, *Pos* is the feature vector of a positive review, n (i.e., 150) is the dimension of a vector. As DBOW and DMM's combination with dimensions 150 is used in the Doc2Vec model learning, the similarity ranges from -2 to 2.

3.7. Negative–Positive Sentimental Statement Datasets

Finally, the NPSS datasets are used to train the STM. In total, from approximately 110 million APRR in 24 categories, we produced over 6 million negative–positive pairs of sentimental statements with resembling semantics. Moreover, using boxplots as shown in Figure 7, we calculate the dispersal of the number of tokens for negative–positive sentimental statements.

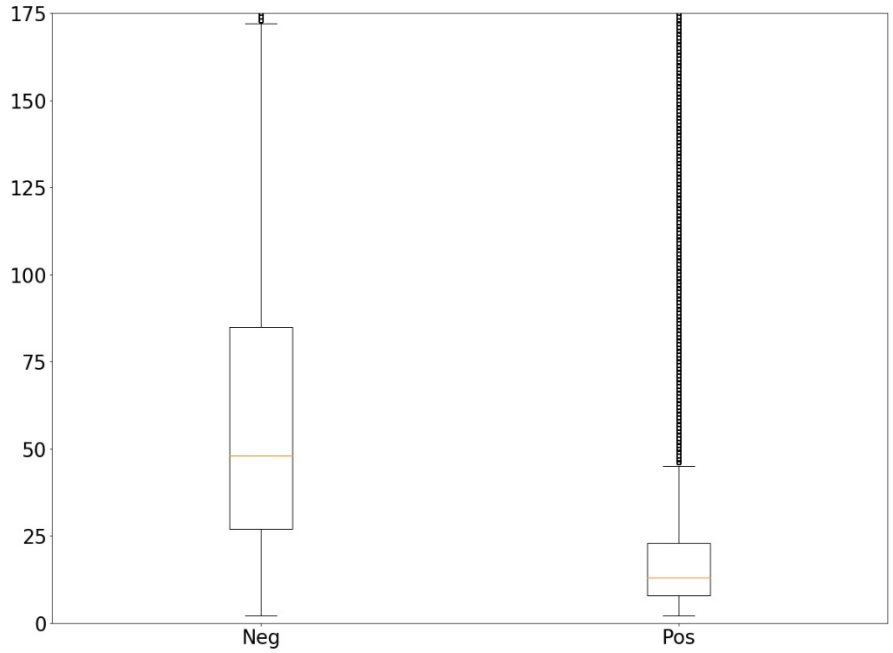

Figure 7. Boxplots of the number of tokens for negative–positive sentimental statements.

4. Translating Sentimental Statements

Recently, in several research areas, deep learning has yielded impressive results. More and more researchers are attempting to use deep learning algorithms to study natural language processing to realize various applications. Recurrent Neural Network (RNN) [23], LSTM [13], and Gated Recurrent Unit (GRU) [24] are widely used neural network types. The state-of-the-art methods used a Seq2seq model to train the encoders and decoders for sequence to sequence translation, based on source and target sequences. For instance, English and Chinese are the source and target sequences, respectively, of an English–Chinese translation application. Similar to the English–Chinese translation model, NSSs are the source sentences, and PSSs are the target sentences of the STM.

In this phase, the STM uses the LSTM-based Seq2seq model for deep learning. In Section 4.1, an LSTM cell is introduced. Then, a sequence generation model Seq2seq used in machine translation is described in Section 4.2, where the appropriate encoders and decoders can also be used in other end-to-end learning applications. Next, the attention mechanism is introduced in Section 4.3, giving the Seq2seq model the ability to judge what features are significant to generate results. Finally, the STM is implemented on the Tensorflow framework, using the 2-layer LSTM-based Seq2seq model with the attention mechanism.

4.1. LSTM

Generally, LSTM can improve the drawback of RNN during training, especially when long sentences are trained. The terms preceding a current word may be ignored for a long paragraph or sentence, allowing the gradient issue to disappear. Several approaches, such as the gradient clipping proposed by Pascanu et al. [25] and using more sophisticated LSTM or GRU neurons, have been proposed to solve this problem. LSTM is used in this paper to interpret sentimental statements in which, compared to RNN, three additional gates (i.e., input, forget, and output gates) are applied to LSTM as shown in Figure 8. These gates are responsible for monitoring previously hidden states, current inputs, and other data that decide what information is discarded or kept.

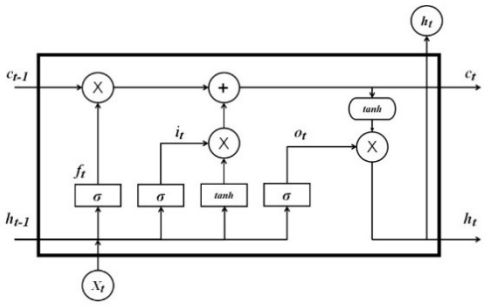

Figure 8. Long Short-Term Memory (LSTM) cell.

The formulas for updating an LSTM cell at time t are as follows:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{2}$$

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{3}$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \tag{4}$$

$$c_t = f_t \circ c_{t-1} + i_t \circ tanh(W_c x_t \circ U_c h_{t-1} + b_c) \tag{5}$$

$$h_t = o_t \circ tanh(c_t) \tag{6}$$

where x_t is an input vector, i_t, f_t, o_t are the activation vectors of input, forget, and output gates, c_t is a cell state vector, h_t is a hidden state vector also known as an LSTM output vector, W and U are weight matrices to be learned during training, and b is a bias vector.

4.2. Seq2seq Model

Seq2seq has been a prevailing model of sequence generation in recent years. Selecting appropriate encoders and decoders is an important step in the end-to-end learning applications. The encoders and decoders can consist of RNN, LSTM, or GRU. The Seq2seq model is given a source sequence and a target sequence, as shown in Figure 9. The source sequence is mapped to the source sequence's tokens in a continuous vector space of fixed dimension by word embedding. The source sequence is then encoded in the encoders and decoded into the target sequence as output in the decoders. For the source sentence "how are you" in the English–Chinese translation model, it enters the encoders via word embedding and is decoded into "你好嗎" when the decoding instruction is received.

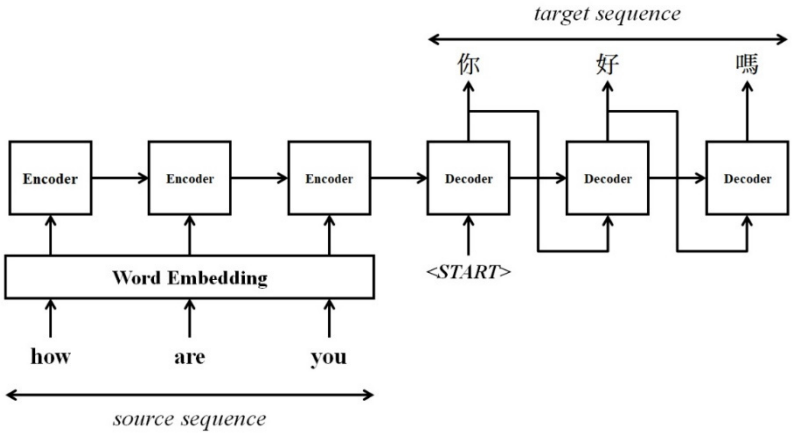

Figure 9. Seq2seq model.

4.3. Attention Mechanism

When it is decoded, words in the Seq2seq model have the same weight. However, the generated sequences' quality may not be good enough if related words and unrelated

words are regarded with the same weight. Currently, the attention mechanism is a common technology and used in deep learning applications. For image classification, the Google Deep-Mind team used it on the RNN model [26]. It was also used by Bahdanau et al. [27] and Luong et al. [28] to translate sentences in machine translation applications. The attention mechanism can determine what features are important for produced outcomes. Words with different weights in a sentence allow the learning of neural networks more versatile in machine translation.

The visualization of the English to German translation attention process, as shown in Figure 10, was provided by Luong et al. [28]. The attention mechanism in the translation pays more attention to words with higher weights to produce sequences. White blocks in the diagram imply higher weights, and black blocks imply lower weights. Here, in the Seq2seq model, we also apply the attention mechanism to produce better PSSs.

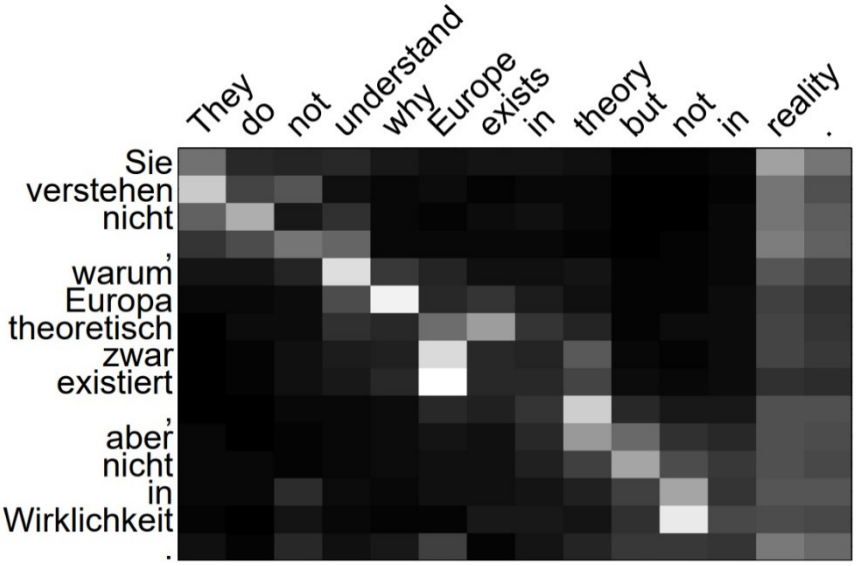

Figure 10. Visualization of the English to German translation attention process [28].

4.4. Sentiment Translation Model

Inspired by the Luong et al. [28] method, the LSTM-based Seq2seq model with the attention mechanism is used in the STM. A 2-layer encoders and decoders structure is used in the Seq2seq model. NSSs are the source sentences, and PSSs are the target sentences of the STM. The trained STM will translate statements from negative sentiment to positive sentiment.

5. Experimental Results

We explain the experimental details in this section, including settings, assessment metrics, and outcomes. First, for the STM learning, the NPSS datasets with 6,294,342 negative–positive pairs of reviews are randomly divided into eight for training, one for validation, and one for the test, respectively. The LSTM-based Seq2seq model with the attention mechanism is employed in the STM, as stated in Section 4. Two evaluation indicators, such as perplexity [29] and bilingual understudy evaluation (BLEU) [30], were used in the experiments to assess the results. Evaluating translated PSSs is somewhat subjective in general, but we still present human assessments on these translated statements.

5.1. Environments

A 2-layer encoders and decoders (or LSTM cells) structure is used in the Seq2seq model. In fact, the three gates in the LSTM cell (i.e., input, forget, and output gates) and *tanh* are neural networks, as shown in Figure 8. For comparison, the number of hidden

cells in these neural networks is set at 512 or 1024. Moreover, the associated parameters used in the experiments are the following:

- Batch size: 256
- Learning rate: 1.0
- Weights initialized from the uniform distribution $[-0.1, 0.1]$
- Optimizer: Stochastic Gradient Descent
- Gradient clipping threshold: 5
- Vocabulary size: 30,000
- Dropout rate: 0.2
- Epochs: 20

In the training, the length of an NSS is at most 172 and the length of a PSS is at most 38. According to negative–positive pairs of reviews collected in the NPSS datasets, the dispersion of the number of tokens for negative–positive sentimental statements using boxplots is shown in Figure 7. We can find that the maximum length of a negative sentiment statement is not more than 172, and the maximum length of a positive sentiment statement is not more than 38. We thought it would not impose loss translation since the STM is trained by using negative–positive pairs of reviews collected in the NPSS datasets. Sentences with the length exceeding the limits are still trained, but encoding or decoding would not process the exceeding parts.

Here, the experiments were implemented in Python, based on the Tensorflow 1.9 library. Case 1 (512 hidden units) was run on GeForce GTX 1060 6GB GPU, whereas Case 2 (1024 hidden units) was run on NVIDIA Tesla K40 12GB GPU.

5.2. Evaluation Indicators

In the experiments, two evaluation indicators evaluate the quality of the STM: (1) perplexity and (2) bilingual evaluation understudy (BLEU).

Perplexity is generally used in natural language processing to measure trained language models [31]. In other applications, language model confusion has been used for domain adaptations such as machine translation [32], spelling correction [33], and text generation, and more. The mathematical form of perplexity is as follows:

$$PP(S) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_1 \cdots w_{i-1})}} \tag{7}$$

where S represents a sentence, N represents the number of words in sentence S, w_i represents the *ith* word in sentence S, P represents the probability of w_i. The lower the perplexity, the higher the probability of combining words in a sentence; in other words, this sentence can be realized by most people, or the generated sequence is more fluent. Therefore, the lower the perplexity value, the better the quality of the STM.

BLEU is a text evaluation algorithm originally used to evaluate the correspondence between machine translation and professional human translation. In general, the closer the machine translation is to the professional human translation, the better the machine translation. Thus, the BLEU algorithm's score can be used as an indicator of STM quality. The higher the BLEU scores, the higher the word similarity to the target sentences. Here, the BLEU scores do not consider grammar correctness, statement polarity (i.e., positive and negative), and sentence semantic similarity. At present, BLEU scores are not only used in machine translation [6] but also used in image caption [34–36], conversational agents [37], etc., to evaluate their models.

5.3. Case Comparisons in Experiment 1

In this experiment, as shown in Figures 11 and 12, we compared the perplexity and BLEU scores of Case 1 (512 hidden units) and Case 2 (1024 hidden units), respectively. We present the perplexity and BLEU scores for the validation and test set every two epochs.

Obviously, lower perplexity can be achieved using only two epochs, as shown in Figure 11, but higher BLEU scores still need more epochs to reach, as shown in Figure 12. Case 2 performs better from these observations than Case 1. Therefore, we only present the experimental findings in the later experiments, using Case 2.

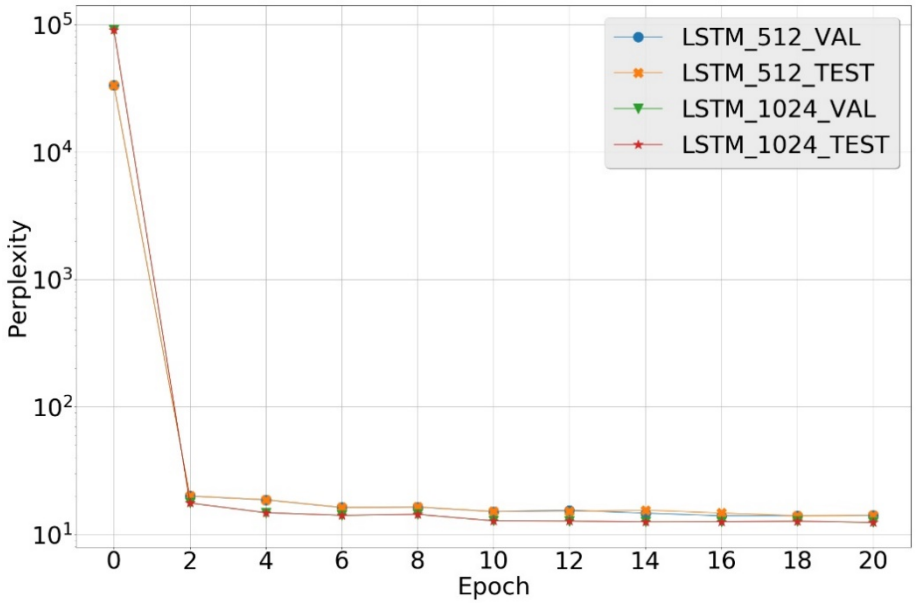

Figure 11. Perplexity of Cases 1 and 2.

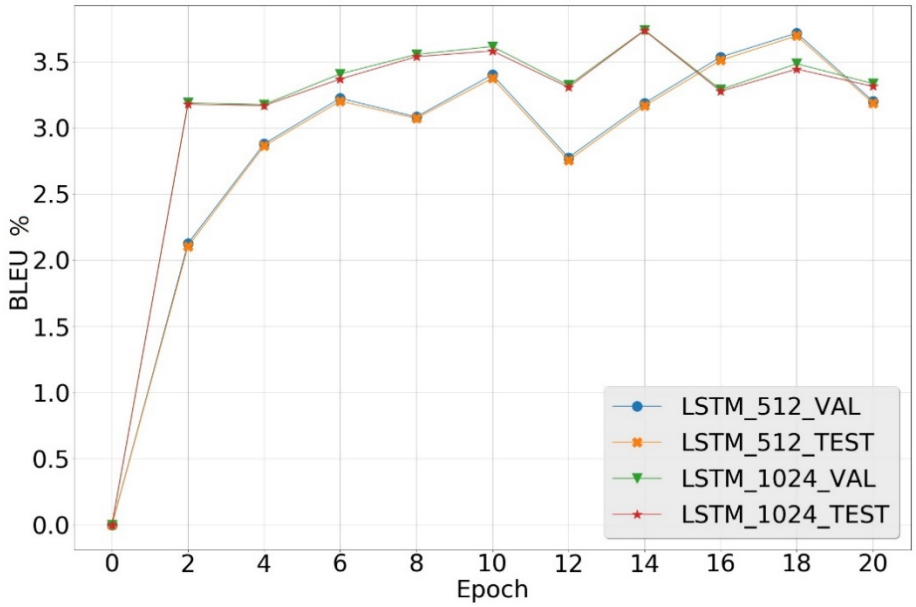

Figure 12. Bilingual understudy evaluation (BLEU) of Cases 1 and 2.

Moreover, as shown in Table 4, we present the best performances of Cases 1 and 2. Generally, applications and datasets strongly affect perplexity and BLEU scores. BLEU scores can reach 37% for machine translation [6], but BLEU scores can only reach 1.9% for conversational agent [37]. While BLEU scores are very low for conversational agent applications, it can still verify that the BLEU scores can truly represent human judgments in the experiments [38].

Table 4. Best performances of Cases 1 and 2.

Case	Dataset	Perplexity	BLEU
LSTM(512)	Validation	13.96	3.71%
	Test	13.98	3.69%
LSTM(1024)	Validation	12.68	3.74%
	Test	12.71	3.73%

5.4. Translated Sentimental Statements in Experiment 1

A few NSSs from the NPSS test set are taken to observe what sentimental statements are translated in the STM in Experiment 1. We show source statements and target statements in the NPSS test set and compare them with the sentimental statements translated by LSTM(512) and LSTM(1024) in Tables 5 and 6. The results comply with the expectations, as shown in Table 5. In Example 1, without any negative sentiments, all the translated statements are the same as the target statement. In Examples 2, 3, and 4, the translated statements, except sentiment factors, are semantically similar to the source statement. Furthermore, Table 6 shows the examples, not in line with the expectations. In Examples 1, 2, and 3, it seems that incorrect target statements train the STM, and this incurs unsatisfied results. In Example 4, although the target statement is OK, the STM cannot infer satisfactory results.

Table 5. Expected translated sentimental statements.

Example 1	
Source	Nice product. Overpriced.
Target	Nice product.
LSTM(512)	Nice product.
LSTM(1024)	Nice product.
Example 2	
Source	This broke after two weeks just junk!
Target	Broke after months of use. I would buy again.
LSTM(512)	Broke after three uses.
LSTM(1024)	Broke after a few weeks but you get what you pay for.
Example 3	
Source	Not very intuitive game play. Maybe I didn't spend enough time on it but it seems slow and boring. Oh just fyi definitely not like a red alert type game.
Target	Fun game maybe not as cool as the first but still enjoyable.
LSTM(512)	I like it but it is a little slow.
LSTM(1024)	I like this game but it is not as intuitive as I thought it would be. Maybe a little slow.
Example 4	
Source	Wobbly and rickety once put together. If the rack wobbles at all the dvd is fall out the back. Looks nice but doesn't t function well. Returning.
Target	Stylish looks nice. Easy to put together.
LSTM(512)	Looks nice and is sturdy.
LSTM(1024)	Looks good but wobbles a lot.

Table 6. Unsatisfied translated sentimental statements.

Example 1	
Source	Horrible product. Broke within a month.
Target	Great product!
LSTM(512)	Good quality and fit.
LSTM(1024)	Good quality and fit right.
Example 2	
Source	Sickening scent. Smells completely different from the dove hairsprays I used previously.
Target	Best smelling hairspray ever!
LSTM(512)	Smells great.
LSTM(1024)	Love the scent.
Example 3	
Source	Cheap and weak. Got weeks out of it.
Target	Got here days later it is good.
LSTM(512)	Got it weeks ago and it works great!
LSTM(1024)	Got it for a good price.
Example 4	
Source	The range is too short for my use. I set up the unit in the living room with the hopes of controlling my cable box from another room. It works as long as the remote is close enough. At about feet, it is intermittent at best.
Target	Range is a little short at around feet but works great closer.
LSTM(512)	Works great. The remote is a little short but it works.
LSTM(1024)	Works great with my directv box.

5.5. Results in Experiment 2

In this experiment, the negative–positive pairs of reviews in the NPSS datasets are sorted in the descending order of cosine similarity, in order to observe the cosine similarity impact on perplexity and BLEU. Here, we use the top 20%, 40%, 60%, 80%, and 100% NPSS datasets to train, validate and test the STM, respectively. Table 7 shows the results using these partial datasets. Obviously, when the cosine similarity of negative–positive pairs of reviews is high, both the perplexity and BLEU scores are better. In other words, using highly similar negative–positive pairs of reviews to train the STM will achieve better results.

Table 7. Performances using partial datasets.

Dataset	Perplexity	BLEU
20%	7.00	9.92%
40%	6.83	8.12%
60%	10.23	5.41%
80%	10.93	4.62%
100%	12.71	3.73%

5.6. Human Assessments

A total of 50 NSSs outside of the NPSS datasets are tested and assessed by five laboratory colleagues for five partial datasets, as stated in Experiment 2, to expose the effects of the STM trained by the NPSS datasets. Human assessments are divided into five levels here, and the examples demonstrating the assessment criteria at five levels are shown in Table 8. Point 1 at level A is that the translated sentimental statement is semantically similar to an NSS and smooth in sentences; point 0.5 at level B is semantically similar and positive, but not what we expect; point 0.25 at level C is not semantically similar, but positive; point 0.25 at level D is semantically similar, but not positive; point 0 at level E is not semantically similar and still negative. The human assessment on these 50 NSSs is

shown in Table 9, based on the assessment criteria. The highest score is obtained in the STM trained by the top 20% NPSS datasets as predicted.

Table 8. Examples demonstrating the assessment criteria at five levels.

Level	Input	Output
A	why not, you stupid bastard.	why not?
B	learning is painful, hate it.	learning so much.
C	you are hard to communicate.	easy to use and easy to hookup.
D	can't sleep. i really hate sleepless nights.	i can't sleep without it.
E	don't bother me.	didn't work for me.

Table 9. Human assessment.

Dataset	A	B	C	D	E	Score
20%	2	9	29	6	4	15.25
40%	0	13	29	2	5	14.25
60%	1	15	25	1	8	15.00
80%	1	11	24	5	9	13.75
100%	1	10	27	3	9	13.50

6. Conclusions

We propose the STM in this paper to translate statements from negative sentiment to positive sentiment. To train the STM, the NPSS datasets of over 6 million negative–positive pairs of sentimental statements with resembling semantics are created based on nearly 110 million APRR obtained from Amazon. For deep learning, the STM uses the LSTM-based Seq2seq model. The application of translating statements from negative sentiment to positive sentiment is challenging, but it is also important to help smooth communication between people. The overall results are satisfactory, but there is still room for improving the NPSS datasets as follows:

1. The Doc2Vec model can find semantically similar statements, but the statements could be contrary semantics. For example, "bad products" and "good products" are similar in semantics, but they are opposite and far from the negative-to-positive translation application.
2. While considering the similarity, the lengths of source statements and target statements should be considered. If their statement lengths are different too much (e.g., the length of a source/target statement is 100/10), it may cause a semantic loss in the translation.

The ground-truth-like NPSS datasets are usually difficult to build automatically. Therefore, the key limitation of developing the application of translating statements from negative sentiment to positive sentiment is how to build the datasets. We think that in the future, the application could be further implemented in two directions. One is to build the NPSS datasets with more similar semantics. The other is to use more sophisticated models to produce text, such as generative adversarial networks or transformers.

Author Contributions: Conceptualization, Y.-F.H.; Funding acquisition, Y.-F.H.; Methodology, Y.-H.L.; Software, Y.-H.L.; Supervision, Y.-F.H.; Validation, Y.-H.L.; Writing—original draft, Y.-H.L.; Writing—review & editing, Y.-F.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data available on request due to restrictions eg privacy or ethical. The data presented in this study are available on request from the corresponding author. The data are not publicly available.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Li, H.; Lu, Z. Deep learning for information retrieval. In Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval, Pisa, Italy, 17–21 July 2016; pp. 1203–1206.
2. Yang, Y.; Yih, W.T.; Meek, C. Wikiqa: A challenge dataset for open-domain question answering. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Lisbon, Portugal, 17–21 September 2015; pp. 2013–2018.
3. Joshi, A.; Fidalgo, E.; Alegre, E.; Fernández-Robles, L. SummCoder: An unsupervised framework for extractive text summarization based on deep auto-encoders. *Expert Syst. Appl.* **2019**, *129*, 200–215. [CrossRef]
4. Hong, J.; Fang, M. *Sentiment Analysis with Deeply Learned Distributed Representations of Variable Length Texts*; Stanford University Report; Stanford University: Stanford, CA, USA, 2015.
5. Huang, Y.F.; Li, Y.H. Sentiment translation model for expressing positive sentimental statements. In Proceedings of the International Conference on Machine Learning and Data Engineering, Taipei, Taiwan, 2–4 December 2019; pp. 79–84.
6. Sutskever, I.; Vinyals, O.; Le, Q.V. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*; The MIT Press: Cambridge, MA, USA, 2014; pp. 3104–3112.
7. Yu, L.; Zhang, W.; Wang, J.; Yu, Y. Seqgan: Sequence generative adversarial nets with policy gradient. In Proceedings of the 31st AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 4–9 February 2017.
8. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems*; The MIT Press: Cambridge, MA, USA, 2017; pp. 5998–6008.
9. He, R.; McAuley, J. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In Proceedings of the 25th International Conference on World Wide Web, Montreal, QC, Canada, 11–15 April 2016; pp. 507–517.
10. McAuley, J.; Targett, C.; Shi, Q.; Van Den Hengel, A. Image-based recommendations on styles and substitutes. In Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, Santiago, Chile, 9–13 August 2015; pp. 43–52.
11. Maas, A.L.; Daly, R.E.; Pham, P.T.; Huang, D.; Ng, A.Y.; Potts, C. Learning word vectors for sentiment analysis. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Portland, OR, USA, 19–24 June 2011; pp. 142–150.
12. Le, Q.; Mikolov, T. Distributed representations of sentences and documents. In Proceedings of the International Conference on Machine Learning, Bejing, China, 22–24 June 2014; pp. 1188–1196.
13. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
14. Hinton, G.E. Learning distributed representations of concepts. In Proceedings of the 8th Annual Conference of the Cognitive Science Society, Amherst, MA, USA, 15–17 August 1986.
15. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient estimation of word representations in vector space. *arXiv* **2013**, arXiv:1301.3781.
16. Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.S.; Dean, J. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*; The MIT Press: Cambridge, MA, USA, 2013; pp. 3111–3119.
17. Mikolov, T.; Yih, W.T.; Zweig, G. Linguistic regularities in continuous space word representations. In Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Atlanta, GA, USA, 9–14 June 2013; pp. 746–751.
18. Pennington, J.; Socher, R.; Manning, C. Glove: Global vectors for word representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, Doha, Qatar, 25–29 October 2014; pp. 1532–1543.
19. Bengio, Y.; Ducharme, R.; Vincent, P.; Jauvin, C. A neural probabilistic language model. *J. Mach. Learn. Res.* **2003**, *3*, 1137–1155.
20. Rehurek, R.; Sojka, P. Software framework for topic modelling with large corpora. In Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks, Valletta, Malta, 22 May 2010.
21. Gensim 3.8.3. Available online: https://pypi.org/project/gensim/ (accessed on 4 May 2020).
22. Pang, B.; Lee, L. Opinion mining and sentiment analysis. *Found. Trends Inf. Retr.* **2008**, *2*, 1–135. [CrossRef]
23. Sutskever, I.; Martens, J.; Hinton, G.E. Generating text with recurrent neural networks. In Proceedings of the 28th International Conference on Machine Learning, Montreal, QC, Canada, 28 June–2 July 2011; pp. 1017–1024.
24. Cho, K.; Van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv* **2014**, arXiv:1406.1078.
25. Pascanu, R.; Mikolov, T.; Bengio, Y. On the difficulty of training recurrent neural networks. In Proceedings of the International Conference on Machine Learning, Atlanta, GA, USA, 16–21 June 2013; pp. 1310–1318.
26. Mnih, V.; Heess, N.; Graves, A. Recurrent models of visual attention. In *Advances in Neural Information Processing Systems*; The MIT Press: Cambridge, MA, USA, 2014; pp. 2204–2212.
27. Bahdanau, D.; Cho, K.; Bengio, Y. Neural machine translation by jointly learning to align and translate. *arXiv* **2014**, arXiv:1409.0473.
28. Luong, M.T.; Pham, H.; Manning, C.D. Effective approaches to attention-based neural machine translation. *arXiv* **2015**, arXiv:1508.04025.
29. Blei, D.M.; Ng, A.Y.; Jordan, M.I. Latent dirichlet allocation. *J. Mach. Learn. Res.* **2003**, *3*, 993–1022.
30. Papineni, K.; Roukos, S.; Ward, T.; Zhu, W.J. BLEU: A method for automatic evaluation of machine translation. In Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, Philadelphia, PA, USA, 6–12 July 2002; pp. 311–318.

31. Chen, S.F.; Goodman, J. An empirical study of smoothing techniques for language modeling. *Comput. Speech Lang.* **1999**, *13*, 359–394. [CrossRef]
32. Qian, X.; Zhong, Z.; Zhou, J. Multimodal machine translation with reinforcement learning. *arXiv* **2018**, arXiv:1805.02356.
33. Mawardi, V.C.; Susanto, N.; Naga, D.S. Spelling correction for text documents in Bahasa Indonesia using finite state automata and Levinshtein distance method. *Proc. MATEC Web Conf.* **2018**, *164*, 01047. [CrossRef]
34. Chen, X.; Lawrence Zitnick, C. Mind's eye: A recurrent visual representation for image caption generation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 2422–2431.
35. Vinyals, O.; Toshev, A.; Bengio, S.; Erhan, D. Show and tell: A neural image caption generator. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 3156–3164.
36. Xu, K.; Ba, J.; Kiros, R.; Cho, K.; Courville, A.; Salakhudinov, R.; Bengio, Y. Show, attend and tell: Neural image caption generation with visual attention. In Proceedings of the International Conference on Machine Learning, Lille, France, 6–11 July 2015; pp. 2048–2057.
37. Li, J.; Galley, M.; Brockett, C.; Spithourakis, G.P.; Gao, J.; Dolan, B. A persona-based neural conversation model. *arXiv* **2016**, arXiv:1603.06155.
38. Galley, M.; Brockett, C.; Sordoni, A.; Ji, Y.; Auli, M.; Quirk, C.; Dolan, B. DeltaBLEU: A discriminative metric for generation tasks with intrinsically diverse targets. *arXiv* **2015**, arXiv:1506.06863.