Adding Domain-free Contexts through Translations for Sentence Classification

Reinald Kim Amplayo[†] and **Kyungjae Lee**[†] and **Jinyeong Yeo**[‡] and **Seung-won Hwang**[†]
[†]Yonsei University, Seoul, South Korea
[‡]Pohang University of Science and Technology, Pohang, South Korea
{rktamplayo, lkj0509, seungwonh}@yonsei.ac.kr jinyeo@postech.edu

Abstract

In sentence classification tasks, additional contexts, such as the neighboring sentences, may improve the accuracy of the classifier. However, such contexts are **domain-dependent** and thus cannot be used for another classification task with an inappropriate domain. In contrast, we propose the use of translated sentences as **domain-free** context that is always available regardless of the domain. We find that naive feature expansion of translations gains only marginal improvements and may decrease the performance of the classifier, due to possible inaccurate translations thus producing noisy sentence vectors. To this end, we present multiple context fixing attachment (MCFA), a series of modules attached to multiple sentence vectors to fix the noise in the vectors using the other sentence vectors as context. We show that our method performs competitively compared to previous models, achieving best classification performance on multiple data sets. We are the first to use translations as domain-free contexts for sentence classification.

1 Introduction

One of the primary tasks in natural language processing (NLP) is sentence classification, where given a sentence (e.g. a sentence of a review) as input, we are tasked to classify it into one of multiple classes (e.g. into positive or negative). This task is important as it is widely used in almost all subareas of NLP such as sentiment classification for sentiment analysis (Pang and Lee, 2007) and question type classification for question answering (Li and Roth, 2002), to name a few. While past methods require feature engineering, recent methods enjoy neural-based methods to automatically encode the sentences into low-dimensional dense vectors (Kim, 2014; Joulin et al., 2017). These vectors are then used as input features to train a classifier. Despite the success of these methods, the major challenge in this task is that extracting features from a single sentence limits the performance.

To overcome this limitation, recent works attempted to augment different kinds of features to the sentence, such as the neighboring sentences (Lin et al., 2015) and the topics of the sentences (Zhao et al., 2017). However, these methods used **domain-dependent** contexts that are only effective when the domain of the task is appropriate. For one thing, neighboring sentences may not be available in some tasks such as question type classification. Moreover, topics inferred using topic models (Steyvers and Griffiths, 2007) may produce less useful topics when the data set is domain-specific such as movie review sentiment classification (Mimno et al., 2011).

In this paper, we propose the usage of translations as compelling and effective **domain-free** contexts, or contexts that are always available no matter what the task domain is. We observe two opportunities when using translations.

First, each language has its own linguistic and cultural characteristics that may contain different signals to effectively classify a specific class. Figure 1 contrasts the sentence vectors of the original English sentences and their Arabic-translated sentences in the question type classification task. A yellow cir-

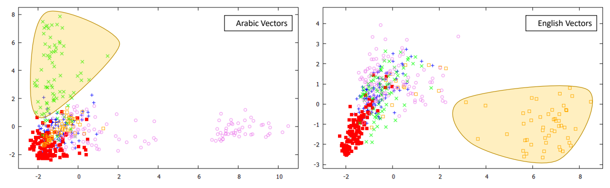


Figure 1: PCA visualizations of unfixed sentence vectors on TREC data set, where each language is effective for a specific class, highlighted using a yellow circle.

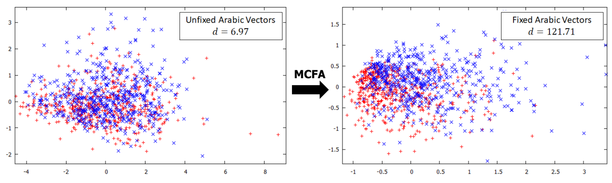


Figure 2: PCA visualizations of unfixed sentence vectors (left) and the corresponding MCFA-fixed vectors (right) on the MR data set. d is the Mahalanobis distance between the two class clusters.

cle signifies a clear separation of a class. For example, the green class, or the numeric question type, is circled in the Arabic space as it is clearly separated from other classes, while such separation cannot be observed in English. Meanwhile, location type questions (in orange) are better classified in English.

Second, the original sentences may include language-specific ambiguity, which may be resolved when presented with its translations. Consider the example English sentence "*The movie is terribly amazing*" for the sentiment classification task. In this case, *terribly* can be used in both positive and negative sense, thus introduces ambiguity in the sentence. When translated to Korean, it becomes "영화는 대단히 훌륭합니다" which means "*The movie is greatly magnificent*", removing the ambiguity.

The above two observations hold only when translations are supported for (nearly) arbitrary language pairs with sufficiently high quality. Thankfully, translation services (e.g. Google Translate[1]) provide readily available translation systems for all pairs of source and target languages. Moreover, recent research on neural machine translation (NMT) (Bahdanau et al., 2014) improved the efficiency and even enabled zero-shot translation (Johnson et al., 2016) of models for languages with no parallel data. This provides an opportunity to leverage on as many languages as possible to any domain, providing a much wider context compared to the limited contexts provided by past studies.

However, despite the maturity of translation, naively concatenating their vectors to the original sentence vector may introduce more noise than signals. The unfixed translation space on the left of Figure 2 shows an example where translation noises make the two classes indistinguishable.

In this paper, we propose a method to mitigate the possible problems when using translated sentences as context based on the following observations. Suppose there are two translated sentences a and b with slight errors. We posit that a can be used to correct b when a is used as a context of b, and vice versa. Revisiting the example above, to fix the vector of the English sentence "*The movie is terribly amazing*", we use the Korean translation to move the vector towards the location where the vector "*The movie is greatly magnificent*" is.

Based on these observations, we present a neural attention-based multiple context fixing attachment (MCFA). MCFA is a series of modules that uses all the sentence vectors (e.g. Arabic, English, Korean, etc.) as context to fix a sentence vector (e.g. Korean). Fixing the vectors is done by selectively moving the vectors to a location in the same vector space that better separates the class, as shown in Figure 2. Noises from translation may cause adverse effects to the vector **itself** (e.g. when a noisy vector is directly used for the task) and **relatively** to other vectors (e.g. when a noisy vector is used to fix another noisy vector). MCFA computes two sentence usability metrics to control the noise when fixing vectors:

- **Self usability** $\rho_i(a)$ weighs the confidence of using sentence a in solving the task.
- **Relative usability** $\rho_r(a, b)$ weighs the confidence of using sentence a in fixing sentence b.

Listed below are the three main strengths of the MCFA attachment:

- **Adaptable to any sentence encoders**: MCFA is attached after encoding the sentence, which makes it widely adaptable to other models.

[1]https://translate.google.com/

- **Expandable to arbitrarily many languages**: MCFA is extensible and improves the accuracy as the number of translated sentences increases.
- **Interpretable vector corrections**: MCFA moves the vectors inside the same space, thus preserves the meaning of vector dimensions.

Results show that a convolutional neural network (CNN) attached with MCFA significantly improves the classification performance of CNN, achieving state of the art performance over multiple data sets. We also show qualitative analysis of our model and provide visualization as to how MCFA fixes vectors based on the sentence usability metrics during the vector correction phase. The code we use in this paper is publicly shared: `http://anonymous.link`.

2 Preliminaries

In this section, we define the problem on how to use translations as additional context. We also discuss the base sentence representation model we use in this paper and two naive extensions to utilize translated sentences as context. Note that since our proposed attachment is orthogonal to the model, other better performing supervised sentence encoding techniques (Ma et al., 2015; Zhang et al., 2016) can also be used. We use a common model for simplicity.

2.1 Problem: Translated sentences as context

In this paper, the ultimate task that we solve is the sentence classification task where given a sentence and a list of classes, one is task to classify which class (e.g. positive or negative sentiment) among the list of classes does the sentence belong. However, the main challenge that we tackle is the task on how to utilize translated sentences as additional context in order to improve the performance of the classifier.

Specifically, the problem states: given the original sentence s, the goal is to use $t_1, t_2, ..., t_n$, or sentences in other languages which are translated from s, as additional context.

2.2 Base model: Convolutional neural network

The base model used is the convolutional neural network (CNN) for sentences (Kim, 2014). It is a simple variation of the original CNN for texts (Collobert et al., 2011) to be used on sentences. Let $\mathbf{x}_i \in \mathbb{R}^d$ be the d-dimensional word vector of the i-th word in a sentence of length n. A convolution operation involves applying a filter matrix $\mathbf{W} \in \mathbb{R}^{h \times d}$ to a window of h words and producing a new feature vector c_i using the equation below, where b is a bias vector and $f(.)$ is a non-linear function.

$$c_i = f([\mathbf{x}_i; ...; \mathbf{x}_{i+h-1}]^\top \mathbf{W} + b) \qquad (1)$$

By doing this on all possible windows of words we produce a feature map $\mathbf{c} = [c_1, c_2, ...]$. We then apply a max-over-time pooling operation (Collobert et al., 2011) over the feature map and take the maximum value as the feature vector of the filter. We do this on all feature vectors and concatenate all the feature vectors to obtain the final feature vector $\mathbf{v}$. We can then use this vector as input features to train a classifier such as logistic regression.

We use CNN to create sentence vectors for all sentences $s, t_1, t_2, ..., t_n$. From here on, we refer to these vectors as $\mathbf{v}_s, \mathbf{v}_{t_1}, \mathbf{v}_{t_2}, ..., \mathbf{v}_{t_n}$, respectively. We refer to them collectively as $\mathbb{V}$.

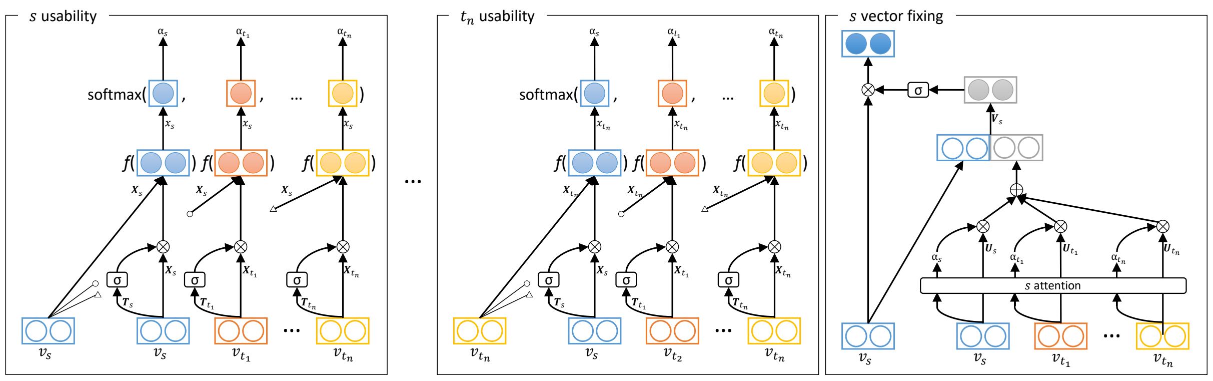
2.3 Baseline 1: Naive concatenation

A simple method in order to use the translated sentences as additional context is to naively concatenate their vectors with the vector of the original sentence. That is, we create a wide vector $\hat{\mathbf{v}} = [\mathbf{v}_s; \mathbf{v}_{t_1}; ...; \mathbf{v}_{t_n}]$, and use this as the input feature vector of the sentence to the classifier.

This method works fine if the translated sentences are translated properly. However, sentences translated using machine translation models usually contain incorrect translation. In effect, this method will have adverse effects on the overall performance of the classifier. This will especially be very evident if the number of additional sentences increases.

2.4 Baseline 2: L2 regularization

In order to alleviate the problems above, we can use L2 regularization to automatically select useful features (Ng, 2004) by weakening the appropriate weights. This is done by inserting the L2 loss of the weight matrix, $\lambda||w||^2$, to the loss function. The main problem of this method occurs when almost all of the weights coming from the vectors of the translated sentence are weakened. This leads to making



(a) Self and relative usability modules

(b) Vector fixing module

Figure 3: Full architecture of the MCFA attachment. An arrow marked with a variable is a matrix multiplication of the vector and the variable. An arrow without a variable simply carries the previous element to the next element. Note that the vector fixing module is also applied to the other contexts, i.e. $t_1, ..., t_n$ and is not shown for conciseness.

the additional context vectors useless and to having a similar performance when there are no additional context. Ultimately, this method does not make use of the full potential of the additional context.

3 Model

To solve the problems of the baselines discussed above, we introduce an attention-based neural multiple context fixing attachment (MCFA), a series of modules attached to the sentence vectors $\mathbb{V}$. MCFA attachment is used to fix the sentence vectors, by slightly modifying the per-dimension values of the vector, before concatenating them into the final feature vector. The sentence vectors are fixed using other sentence vectors as context (e.g. $\mathbf{v}_{t_1}$ is fixed using $\mathbf{v}_s, \mathbf{v}_{t_2}, ..., \mathbf{v}_{t_n}$). This results to moving the vectors in the same vector space. The full architecture is shown in Figure 3. In the next sections, we discuss the subparts of MCFA attachment: the self usability module, the relative usability module, and the vector fixing module.

3.1 Self usability module

To fix a source sentence vector[2], we use the other sentence vectors as guide to know which dimensions to fix and to what extent do we need to fix them. However, other vectors might also contain errors which may reflect to the fixing of the source sentence vector. In order to cope with this, we introduce self usability modules. A self usability module contains the **self usability** of the vector $\rho_i(a)$, which measures how confident sentence a is for the task at hand. For example, an ambiguous sentence (e.g. "*The movie is terribly amazing*") may receive a low self usability, while a clear and definite sentence (e.g. "*The movie is very good*") may receive a high self usability.

Mathematically, we calculate the self usability of the vector $\mathbf{v}_i$ of sentence i, denoted as $\rho_i(\mathbf{v}_i)$, using the equation below, where $\mathbf{T}_i \in \mathbb{R}^{d \times 1}$ is a matrix to be learned. The produced value is a single real number from 0 to 1.

$$\rho_i(\mathbf{v}_i) = \sigma(\mathbf{v}_i^\top \mathbf{T}_i) \qquad (2)$$

We pre-calculate the self usability of all sentence vectors $\mathbf{v}_i \in \mathbb{V}$. These are used in the next module, the relative usability module.

3.2 Relative usability module

Relative usability $\rho_r(a, b)$ measures how useful can a be when fixing b, relative to other sentences. There are two main differences between $\rho_i(a)$ and $\rho_r(a, b)$. First, $\rho_i(a)$ is calculated before a knows about b while $\rho_r(a, b)$ is calculated when a knows about b. Second, $\rho_r(a, b)$ can be low even though $\rho_i(a)$ is not. This means that a is not able to help in fixing

[2]Hereon, we say that $\mathbf{v}_k$ is a *source sentence vector* if $\mathbf{v}_k$ is the current vector to be fixed.

the wrong information in b. Here, we extend the additive attention module (Bahdanau et al., 2014) and use it as a method to calculate the relative usability of two sentences of different languages. To better visualize the original attention mechanism, we present the equations below.

$$e_i = u^\top tanh(s^\top \mathbf{W} + t_i^\top \mathbf{U}) \tag{3}$$

$$\alpha_i = \frac{exp(e_i)}{\sum_{j \in T} exp(e_j)} \tag{4}$$

One major challenge in using the attention mechanism in our problem is that the sentence vectors do not belong to the same vector space. Moreover, one characteristic of our problem is that the sentence vectors can be both a source and a context vector (e.g. $\mathbf{v}_s$ can be both s and t_i in Equation 3). Because of these challenges, we cannot directly use the additive attention module. We extend the module such that (1) each sentence vector $\mathbf{v}_k$ has its own projection matrix $\mathbf{X}_k \in \mathbb{R}^{d \times d}$, and (2) each projection matrix $\mathbf{X}_k$ can be used as projection matrix of both the source (e.g. when sentence k is the current source) and the context vectors. Finally, we incorporate the self usability function $\rho_i(\mathbf{v}_k)$ to reflect the self usability of a sentence. Ultimately, the relative usability denoted as $\rho_r(\mathbf{v}_i, \mathbf{v}_j)$ is calculated using the equations below, where $\times$ is the multiplication of a vector and a scalar through broadcasting.

$$e(\mathbf{v}_i, \mathbf{v}_j) = x^\top tanh(\mathbf{v}_i^\top \mathbf{X}_i + \mathbf{v}_j^\top \mathbf{X}_j \times \rho_i(\mathbf{v}_j)) \tag{5}$$

$$\rho_r(\mathbf{v}_i, \mathbf{v}_j) = \frac{exp(e(\mathbf{v}_i, \mathbf{v}_j))}{\sum_{\mathbf{v}_k \in \mathbb{V}} exp(e(\mathbf{v}_i, \mathbf{v}_k))} \tag{6}$$

3.3 Vector fixing module

The final module is the vector fixing module, where we apply the attention weights to the sentence vectors and create an integrated context vector. We then use this vector alongside with the source sentence vector to create a weighted gate vector. The weighted gate vector is used to determine to what extent should a dimension of the source sentence vector be fixed.

The common way to apply the attention weights to the context vectors and create an integrated context vector c_i is to directly do weighted sum of all the context vectors. However, this is not possible because the context vectors are not on the same space. Thus, we use a projection matrix $\mathbf{U}_k \in \mathbb{R}^{d \times d}$ to linearly project the sentence vector $\mathbf{v}_k$ to transform the sentence vectors into a common vector space. The integrated context vector c_i is then calculated using the equation below.

$$c_i = \sum_{\mathbf{v}_k \in \mathbb{V}} \alpha_{ik} \mathbf{v}_k^\top \mathbf{U}_k \tag{7}$$

Finally, we construct a weighted gate vector w_k and use it to fix the source sentence vectors using the equations below, where $\mathbf{V}_k \in \mathbb{R}^{2d \times d}$ is a trainable parameter and $\otimes$ is the element-wise multiplication procedure. The weighted gate vector is a vector of real numbers between 0 and 1 to modify the intensity of per-dimension values of the sentence vector. This causes the vector to move in the same vector space towards the correct direction.

$$w_k = \sigma([\mathbf{v_k}; c_k]^\top \mathbf{V}_k) \tag{8}$$

$$\hat{\mathbf{v}}_k = \mathbf{v}_k \otimes w_k \tag{9}$$

An alternative approach to do vector correction is using a residual-style correction, where instead of multiplying a gate vector, a residual vector (He et al., 2016) is added to the original vector. However, this approach makes the correction not interpretable; it is hard to explain what does adding a value to a specific dimension mean. One major advantage of MCFA is that the corrections in the vectors are interpretable; the weights in the gate vector correspond to the importance of the per-dimension features of the vector.

The fixed vectors $\hat{\mathbf{v_s}}, ..., \hat{\mathbf{v_{t_n}}}$ are then concatenated and fed directly as an input vector to the logistic regression classifier for training.

4 Experiments

4.1 Experimental setting

We test our model on four different data sets as listed below and summarized in Table 1.

- **MR**[3] (Pang and Lee, 2005): Movie reviews where each data instance is a sentence. The task is to classify whether the sentence has positive or negative sentiment.

[3]`https://www.cs.cornell.edu/people/pabo/movie-review-data/`

Data set	c	$\|w\|$	M	*Test*
MR	2	20	10662	CV
SUBJ	2	19	10000	CV
CR	2	23	3775	CV
TREC	6	10	5952	500

Table 1: Statistics of the four data sets used in this paper. c: number of target classes. $|w|$: average number of words. M: number of data instances. *Test*: size of the test data, if available. If not, we use 10-fold cross validation (marked as CV).

- **SUBJ** (Pang and Lee, 2004): Subjectivity data set where the task is to classify whether the sentence is subjective or objective.
- **CR**[4] (Hu and Liu, 2004): Customer reviews where each data instance is a review of a certain product. The task is to classify whether the sentiment is positive or negative.
- **TREC**[5] (Li and Roth, 2002): TREC question data set where given a question, the task is to classify what type of question is it (total of six question types).

All our data sets are in English. For the additional contexts, we use ten other languages, selected based on their diversity and their performance on prior experiments: Arabic, Finnish, French, Italian, Korean, Mongolian, Norwegian, Polish, Russian, and Ukranian. We translate the data sets using Google Translate[6]. Preprocessing is done using the polyglot library[7]. We experiment on using only one additional context ($N = 1$) and using all ten languages at once ($N = 10$). For $N = 1$, we only show the accuracy of the best classifier for conciseness.

For our CNN, we use rectified linear units and three filters with different window sizes $h = 3, 4, 5$ with 100 feature maps each. For the final sentence vector, we concatenate the feature maps to get a 300-dimension vector. We use dropout (Srivastava et al., 2014) on all non-linear connections with a dropout rate of 0.5. We also use an l_2 constraint (Hinton et al., 2012) of 3, following (Kim, 2014) for accurate comparisons. We use pre-trained vectors provided by FastText[8] (Bojanowski et al., 2016) for all our data sets and their corresponding additional context. During training, we use mini-batch size of 50. Training is done via stochastic gradient descent over shuffled mini-batches with the Adadelta update rule. We perform early stopping using 10% of the training set as the development set, randomly selected.

We present several competing models, listed below to compare the performance of our model. We do not show results from RNN models because they were shown to be less effective in sentence classification (Zhang et al., 2016). For models with additional context, we further use an ensemble classification model by averaging the class probability scores generated by the multiple variants (in our model's case, $N = 1$ and $N = 10$ models), following (Zhao et al., 2017).

- CNN-based models: Aside from the base model (**CNN**) (Kim, 2014), we use two other CNN-based models. Dependency-based CNN (**Dep-CNN**) (Ma et al., 2015) parses the sentences first and does convolution on ancestor paths. Dependency-sensitivity CNN (**DSCNN**) (Zhang et al., 2016) uses LSTM to capture dependency information within each sentence.
- AdaSent: **AdaSent** (Zhao et al., 2015) adopts a hierarchical structure, where consecutive levels are connected through gated recursive composition of adjacent segments, and feeds the hierarchy as a multi-scale representation through a gating network.
- CNN+topics: Topic-aware Convolutional Neural Network (**TopCNN**) (Zhao et al., 2017) uses topics as additional contexts and changes the CNN architecture. TopCNN uses two types of topics: word-specific topic and sentence-specific topic.
- CNN+translations (naive): **CNN+B1** and **CNN+B2** are the two baselines presented in this paper, which naively concatenate the sentence vectors of the original sentence and the

[4]http://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html

[5]http://cogcomp.cs.illinois.edu/Data/QA/QC/

[6]https://translate.google.com

[7]https://pypi.python.org/pypi/polyglot

[8]https://github.com/facebookresearch/fastText/blob/master/pretrained-vectors.md

Model	**MR**			**SUBJ**			**CR**			**TREC**		
CNN	81.5			93.4			85.0			93.6		
Dep-CNN	81.9			-			-			95.4		
DSCNN	82.2			93.2			-			**95.6**		
AdaSent	**83.1**			**95.5**			86.3			92.4		
C = Topic	word	sent	ens	word	sent	ens	word	sent	ens	word	sent	ens
TopCNN	81.7 (+0.2)	81.3 (-0.2)	83.0 (+1.5)	93.4 (+0.0)	93.4 (+0.0)	95.0 (+1.6)	84.9 (-0.1)	84.8 (-0.2)	**86.4 (+1.4)**	92.5 (-1.1)	92.0 (-1.6)	94.0 (+0.4)
C = Trans	N=1	N=10	ens	N=1	N=10	ens	N=1	N=10	ens	N=1	N=10	ens
CNN+B1	81.9 (+0.4)	81.4 (-0.1)	82.6 (+1.1)	94.6 (+1.2)	93.8 (+0.4)	94.9 (+1.5)	86.2 (+1.2)	85.9 (+0.9)	86.7 (+1.7)	95.4 (+1.8)	95.0 (+1.4)	96.4 (+3.0)
CNN+B2	82.1 (+0.6)	82.1 (+0.6)	82.2 (+0.7)	94.6 (+1.2)	94.0 (+0.6)	94.8 (+1.4)	86.1 (+1.1)	86.3 (+1.3)	86.6 (+1.6)	95.4 (+1.8)	95.2 (+1.6)	96.4 (+3.0)
CNN+MCFA	82.3 (+0.8)	82.7 (+1.2)	**83.2 (+1.7)**	94.7 (+1.3)	94.8 (+1.4)	**95.2 (+1.8)**	87.6 (+2.6)	88.6 (+3.6)	**89.4 (+4.4)**	95.4 (+1.8)	96.0 (+2.4)	**96.8 (+3.4)**

Table 2: Classification accuracies of competing models. **C** refers to the additional context. Accuracies colored red are accuracies that perform worse than CNN. Previous state of the art results and the results of our best model are **bold-faced**. The winning result is underlined. The number inside the parenthesis indicates the increase from the base model, CNN.

translated sentences, and additionally uses L2 regularization for feature selection.

4.2 Results and discussion

Classification results We report the classification accuracy of the competing models in Table 2. We show that CNN+MCFA achieves state of the art performance on three of the four data sets and performs competitively on one data set. When $N = 1$, MCFA increases the performance of a normal CNN from 85.0 to 87.6, beating the current state of the art on the CR data set. When $N = 10$, MCFA additionally beats the state of the art on the TREC data set. Finally, our ensemble classifier additionally outperforms all competing models on the MR data set. We emphasize that we only use the basic CNN as our sentence encoder for our experiments, yet still achieve state of the art performance on most data sets. Hence, MCFA is successful in effectively using translations as additional context to improve the performance of the classifier.

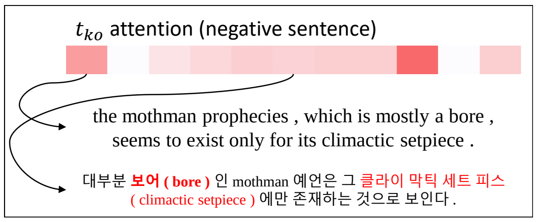
We compare our model (CNN+MCFA) and the baselines discussed above (CNN+B1, CNN+B2). On all settings, our model outperforms the baselines. When $N = 10$, the performance of our model increases over the performance when $N = 1$, however the performance of CNN+B1 decreases when compared to the performance when $N = 1$. We also show the accuracies of the worst classifiers when

Model	**MR**	**SUBJ**	**CR**	**TREC**
CNN	81.5	93.4	85.0	93.6
CNN+B1	81.4	94.2	83.8	93.0
CNN+B2	81.7	94.2	84.0	93.2
CNN+MCFA	81.8	94.4	85.8	94.2

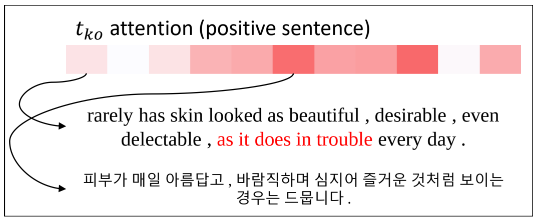
Table 3: Accuracies of the worst CNN+translation classifiers when $N = 1$. Accuracies less than CNN accuracies are highlighted in red.

$N = 1$ in Table 3. On all data sets except SUBJ, the accuracy of CNN+B1 decreases from the base CNN accuracy, while the accuracy of our model always improves from the base CNN accuracy. We explain that these phenomena are due to the increase of noisy vectors due to wrong translations. This is resolved by CNN+B2 by applying L2 regularization, however the increase in performance is marginal.

We also compare two different kinds of additional context: topics (TopCNN) and translations (CNN+B1, CNN+B2, CNN+MCFA). Overall, we conclude that translations are better additional contexts than topics. When using a single context (i.e. TopCNN$_{\text{word}}$, TopCNN$_{\text{sent}}$, and our models when $N = 1$), translations always outperform topics even when using the baseline methods. Using topics as additional context also decreases the performance of the CNN classifier on most data sets, giving an adverse effect to the CNN classifier.


t_{ko} attention (negative sentence)
the mothman prophecies , which is mostly a bore , seems to exist only for its climactic setpiece .
대부분 보어 (bore) 인 mothman 예언은 그 클라이 막틱 세트 피스 (climactic setpiece) 에만 존재하는 것으로 보인다 .

(a) Example where English attention weight is larger


t_{ko} attention (positive sentence)
rarely has skin looked as beautiful , desirable , even delectable , as it does in trouble every day .
피부가 매일 아름답고 , 바람직하며 심지어 즐거운 것처럼 보이는 경우는 드뭅니다 .

(b) Example where Korean attention weight is larger

Figure 4: Attention weights of example Korean sentences from the MR data set. The red color fill represents the attention weights given to each sentence. The darker the fill, the larger the attention weight.

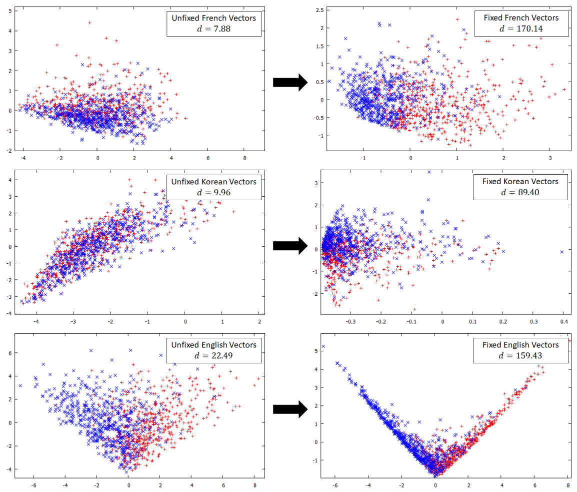


Figure 5: PCA visualization of unfixed (left) and fixed (right) vectors of the MR data set. d is the Mahalanobis distance between two class clusters.

5 Model interpretation

In the next sections, we look at the different parts of MCFA and provide visualizations for interpretation.

We first provide examples shown in Table 4 on how the self usability module determines the score of sentences. In the first example, it is hard to classify whether the translated sentence is positive or negative, thus it is given a low self usability score. In the second example, although the sentence contains mistranslations, these are minimal and may actually help the classifier by telling it that *thirst for violence* is not a negative phrase. Thus, it is given a high self usability score.

Figure 4 shows two data instance examples where we show the attention weights given to the other contexts when fixing the Korean sentence. That is, the larger the attention weight is, the more the context is used to fix the Korean sentence. In the first example, the Korean sentence contains translation errors; especially, the words *bore* and *climactic setpiece* were not translated but rather were only spelled using the Korean alphabet. In this example, the English attention weight is larger than the Korean attention weight. In the second example, the Korean sentence correctly translates all parts of the English sentence, except for the phrase *as it does in trouble*. However, this phrase is actually not necessary to classify the sentence correctly, and may induce possible vague classification because of the word *trouble*. Thus, in this case, the Korean attention weight is larger than the English attention weight.

We show the effectiveness of the vector fixing module in fixing the context vectors. Figure 5 shows the PCA visualization of the unfixed and the fixed vectors of four different languages. In the first example, the unfixed sentence vectors are mostly in the middle of the vector space, making it hard to draw a boundary between the two examples. After the fixing, the boundary is much clearer. In the second example, it is impossible to distinguish the classes as the vectors are overlapping with each other. After the fix, the blue-class and red-class vectors moved to the opposite sides, making it easier to tell and differentiate the two classes. Finally, we also show the English sentence vectors in the last example. Even without fixing the unfixed English sentence vectors, it is already easy to distinguish both classes. After the fix, the sentence vectors in the middle of the space are moved, making the distinction more obvious and clearer. We also provide quantitative evidence by showing that the Mahalanobis distance between the two classes in the fixed vectors are significantly farther than that of the unfixed vectors.

We also show two examples sentences from En-

Original sentence: *skip this turd and pick your nose instead because you're sure to get more out of the latter experience .*
Korean translation: *후자의 경험에서 더 많은 것을 얻으려면 이 웅덩이를 건너 뛰고 코를 골라야합니다.*
Human re-translation: *In order to get more from the latter experience , you need to skip this puddle and choose your nose .*
Self Usability: 0.3958

(a) Low self usability example

Original sentence: *michael moore's latest documentary about america's thirst for violence is his best film yet . . .*
Korean translation: *마이클 무어 (Michael Moore) 의 최근 미국 다큐멘터리 " 폭력 장면 " 은 그의 최고의 영화 다 . . .*
Human re-translation: *Michael Moore's latest American documentary " Violent Scene " is his best film yet . . .*
Self Usability: 1.0000

(b) High self usability example

Table 4: Two examples of self usability of Korean sentences from the MR data set. Texts colored in red are mistranslated texts.

Sentence	may take its sweet time to get wherever it's going, but if you have the patience for it, you won't feel like it's wasted yours.
NN (Unfixed)	you know that ten bucks you'd spend on a ticket? just send it to cranky. we don't get paid enough to sit through crap like this.
NN (Fixed)	what might have been readily dismissed as the tiresome rant of an aging filmmaker still thumbing his nose at convention takes a surprising, subtle turn at the midway point.
Sentence	every nanosecond of the new guy reminds that you could be doing something else more pleasurable. like scrubbing the toilet. emptying rat traps. or doing last year's taxes with your ex-wife.
NN (Unfixed)	in the new release of cinema paradiso, the tale has turned from sweet to bittersweet, and when the tears come during that final, beautiful scene, they finally feel absolutely earned.
NN (Fixed)	after scenes of this nonsense, you'll be wistful for the testosterone-charged wizardry of jerry bruckheimer productions, especially because half past dead is like the rock on wal-mart budget.

Table 5: Two example sentences, from English (first) and Korean (second) vector spaces, and their nearest neighbors (NN) on both the unfixed and fixed vector spaces. We only show the original English sentences for the Korean example for conciseness.

glish and Korean vector spaces and their corresponding nearest neighbors on both the unfixed and fixed vector spaces in Table 5. In the first example, the unfixed vector focuses on the meaning of *"wasted yours"* in the sentence, which puts it near sentences regarding wasted time or money. After fixing, the sentence vector focuses its meaning on the slow yet worth-the-wait pace of the movie, thus moving it closer to the correct vectors. In the second example, all three sentences have highly descriptive tones, however, the nearest neighbor on the fixed space is hyperbolically negative, comparing the movie to a description unrelated to the movie itself.

Additionally, we look at the performance of all the languages when independently added as an additional context to CNN+MCFA when $N = 1$. For each dataset, we rank the accuracies of the models with different languages and report the results in Table 6. We find two observations interesting and require further investigation. First, the performance of the added language depends on the kind of task and dataset. In the MR and CR datasets where the task is sentiment classification given review texts, the Finnish language model (colored red) garners better accuracy compared to other languages, however it does not perform as well in the TREC dataset

Rank	MR	CR	SUBJ	TREC
1	**FI**	UK	IT	**FR**
2	UK	**FI**	PL	RU
3	IT	KO	AR	MN
4	**NO**	PL	**FI**	PL
5	AR	**NO**	**FR**	**NO**
6	RU	IT	MN	AR
7	KO	RU	**NO**	UK
8	**FR**	**FR**	RU	KO
9	MN	AR	UK	**FI**
10	PL	MN	KO	IT

Table 6: Ranking of the accuracies of a language when used as an additional context for CNN+MCFA when $N = 1$ on multiple datasets.

where the task is classifying question types. Similar phenomenon is observed in the French language model (colored blue) where it performs better in the TREC dataset but performs relatively worse in the MR and CR datasets. This may be due to two things: the characteristic of translation data and the nature of the language itself. Second, we notice that the Norwegian language model (colored green) performs moderately on all datasets. We therefore recommend the Norwegian language as additional context when the task and the given dataset are different from the datasets used in the paper. Ultimately, these findings require further analysis and we leave this in the future work.

6 Related work

One way to improve the performance of a sentence classifier is to introduce new context. Common and obvious kinds of context are the neighboring sentences of the sentence (Lin et al., 2015), and the document where the sentence belongs (Huang et al., 2012). Another possible type of context is the topic (Amiri et al., 2016) of the sentence. Topics of the words in the sentence induced by Latent Dirichlet Allocation (LDA) topic model (Blei, 2012), combined with sentence topics, were also used as contexts (Zhao et al., 2017). In this paper, we introduce yet another type of additional context, sentence translations, which to the best of our knowledge have not been used previously.

Sentence encoders trained from neural machine translation (NMT) systems were also used for transfer learning. The effects of transfer learning using NMT encoders from a variety of source domains were first studied to semantic similarity tasks (Hill et al., 2016). (Hill et al., 2017) demonstrated that fixed-length sentence vectors from NMT encoders outperform sentence vectors from monolingual encoders on semantic similarity tasks. Recent work used representation of each word in the sentence to create a sentence representation suitable for multiple NLP tasks (McCann et al., 2017). Our work shares the commonality of using NMT for another task, but instead of using NMT to encode our sentences, we use it to translate the sentences into new contexts.

Increasing the number of data instances of the training set has also been explored to improve the performance of a classifier. Past methods include incremental learning where the model is increasingly trained using automatically labelled data (Yan et al., 2009). Recent methods include the usage of thesaurus (Zhang et al., 2015), paraphrases (Fu et al., 2014), and context re-ordering (Pan et al., 2016). These simple variation techniques are preferred because they are found to be very effective despite their simplicity. Our work similarly augments training data, not by adding data instances (vertical augmentation), but rather by adding more context (horizontal augmentation). Though the paraphrase of p can be alternatively used as an augmented context, this could not leverage the added semantics coming from another language, as discussed in Section 1.

7 Conclusion

This paper investigates the use of translations as better additional contexts for sentence classification. To answer the problem on mistranslations, we propose a neural attention-based solution called multiple context fixing attachment (MCFA) to fix the context vectors using other context vectors. We show that our method improves the classification performance and achieves state-of-the-art performance on multiple data sets. We also provide qualitative analysis on how our method is effective on fixing noisy context vectors and how languages help differently depending on the dataset. In our future work, we plan to use and extend our model to other complex NLP tasks such as language inference and question answering.

References

Hadi Amiri, Philip Resnik, Jordan Boyd-Graber, and Hal Daumé III. 2016. Learning text pair similarity with context-sensitive autoencoders. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1882–1892.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

David M Blei. 2012. Probabilistic topic models. *Communications of the ACM*, 55(4):77–84.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.

Guohong Fu, Yu He, Jiaying Song, and Chaoyue Wang. 2014. Improving chinese sentence polarity classification via opinion paraphrasing. *CLP 2014*, page 35.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning distributed representations of sentences from unlabelled data. *arXiv preprint arXiv:1602.03483*.

Felix Hill, Kyunghyun Cho, Sébastien Jean, and Yoshua Bengio. 2017. The representational geometry of word meanings acquired by neural machine translation models. *Machine Translation*, pages 1–16.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM.

Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 873–882. Association for Computational Linguistics.

Melvin Johnson, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. 2016. Google's multilingual neural machine translation system: enabling zero-shot translation. *arXiv preprint arXiv:1611.04558*.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *EACL*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics.

Rui Lin, Shujie Liu, Muyun Yang, Mu Li, Ming Zhou, and Sheng Li. 2015. Hierarchical recurrent neural network for document modeling. In *EMNLP*, pages 899–907.

Mingbo Ma, Liang Huang, Bing Xiang, and Bowen Zhou. 2015. Dependency-based convolutional neural networks for sentence embedding. *arXiv preprint arXiv:1507.01839*.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. *arXiv preprint arXiv:1708.00107*.

David Mimno, Hanna M Wallach, Edmund Talley, Miriam Leenders, and Andrew McCallum. 2011. Optimizing semantic coherence in topic models. In *Proceedings of the conference on empirical methods in natural language processing*, pages 262–272. Association for Computational Linguistics.

Andrew Y Ng. 2004. Feature selection, l 1 vs. l 2 regularization, and rotational invariance. In *Proceedings of the twenty-first international conference on Machine learning*, page 78. ACM.

Da Pan, Jiaying Song, and Guohong Fu. 2016. Expanding corpora for chinese polarity classification via opinion paraphrase generation. In *International Conference of Young Computer Scientists, Engineers and Educators*, pages 362–373. Springer.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd annual meeting on Association for Computational Linguistics*, page 271. Association for Computational Linguistics.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 115–124. Association for Computational Linguistics.

Bo Pang and Lillian Lee. 2007. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2:1–135.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958.

Mark Steyvers and Tom Griffiths. 2007. Probabilistic topic models. *Handbook of latent semantic analysis*, 427(7):424–440.

Hongcan Yan, Chen Lin, and Bicheng Li. 2009. A svm-based text classification method with ssk-means clustering algorithm. In *Artificial Intelligence and Computational Intelligence, 2009. AICI'09. International Conference on*, volume 2, pages 379–383. IEEE.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

Rui Zhang, Honglak Lee, and Dragomir Radev. 2016. Dependency sensitive convolutional neural networks for modeling sentences and documents. *arXiv preprint arXiv:1611.02361*.

Han Zhao, Zhengdong Lu, and Pascal Poupart. 2015. Self-adaptive hierarchical sentence model. In *IJCAI*, pages 4069–4076.

Rui Zhao, Kezhi Mao, Rui Zhao, and Kezhi Mao. 2017. Topic-aware deep compositional models for sentence classification. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 25(2):248–260.