Multi-Label Playlist Classification Using Convolutional Neural Network

Guan-Hua Wang[1], Chia-Hao Chung[1], Yian Chen[2], and Homer Chen[1]
[1]National Taiwan University
E-mail: {r05942102, f03942038, homer}@ntu.edu.tw
[2]KKBOX Inc.
Email: annchen@kkbox.com

***Abstract*—With the popularization of music streaming services, millions of songs can be accessed easily. The effectiveness of music access can be further enhanced by making the labels of a music playlist more indicative of the theme of the playlist. However, manually classifying playlists is laborious and often requires domain knowledge. In this paper, we propose a novel multi-label model for playlist classification based on a convolutional neural network. The network is trained in an end-to-end manner to jointly learn song embedding and convolutional filters without the need of feature extraction from audio signals. Specifically, the song embedding vectors are concatenated as a matrix to represent a playlist, and the convolutional filters for playlist classification are applied to the playlist matrix. We also propose two augmentation techniques to prevent over-fitting of playlist data and to improve the training of the proposed model. Experimental results show that the proposed model performs significantly better than the support vector machine and *k*-nearest neighbors models.**

I. INTRODUCTION

The advancement of music streaming services enables users to access almost any song ever recorded. In the face of such vast and diverse music collections, users may often find it difficult to select music [1]. Music playlist, which is a sequence of songs to be listened to as a group, can help alleviate the difficulty and allow users to discover music easily [2]–[4]. Typically, each playlist has a theme reflecting some aspects, such as artist, genre, specific event, or mood [3], of the playlist. The use of playlist greatly enhances the effectiveness and efficiency of music discovery and leads to enjoyable music listening experiences. For example, it helps songs to be collected in an organized fashion, allows user preference to be matched, and enriches music sharing over social networks.

Over the years, the use of playlist for music streaming has become popular [2]–[6], and the need for playlist labeling and classification has grown significantly. However, manual classification of playlists is a laborious job and often requires domain knowledge. While there is abundant work on automatic music classification in the field of music information retrieval [7]–[9], little attention has been paid to automatic playlist classification, probably because playlist labels are hard to collect. The fact that playlist classification needs to handle multiple tracks also makes it a more complex problem than automatic music classification, which only needs to consider a single track at a time.

A simple approach to playlist classification is to formulate each playlist as a k-hot vector, where the length of the vector is the total number of tracks, and each element of the vector represents a track. An element is set to one if the corresponding track exists in the playlist, otherwise, it is set to zero. Then a classifier, such as the *k*-nearest neighbors model, can be applied to the k-hot vectors to measure the similarity between the playlists by the number of identical tracks. However, this approach has limited use in practice because playlists hardly have a significant number of identical tracks. Such sparsity makes it difficult to find optimal decision boundaries for the classifier [10].

In this work, we aim to develop a convolutional neural network-based multi-label playlist classification model, where each input playlist is represented by a matrix consisting of the embedding vectors of the songs in the playlist. The model is trained in an end-to-end manner to jointly learn the song embedding vectors and convolutional filters without the need of audio feature extraction. Moreover, we design our model using both 2-D and 1-D convolutional filters to achieve a higher classification performance than a model using only 1-D convolutional filters.

II. RELATED WORK

Most previous studies related music playlist focus on the analysis of existing playlists [3]–[5] or the creation of new playlists [2], [6]. A common idea underlying these studies is that songs in the same playlist should be similar to each other in some aspects. Keeping this idea in mind, we perform in this work a multi-label playlist classification to deepen the understanding of music playlist and to expand the field of music information retrieval.

Convolutional neural network (CNN) has been found useful for a wide range of text classification tasks, such as sentence classification, election classification, and sentiment analysis [11]–[14]. In these tasks, each word can be projected into an embedding space and represented by a fixed-dimension vector. The word embedding vectors can be learned either jointly with the classification model [11]–[13] or separately (in a pre-train manner) using unsupervised methods such as word2vec [14], [15]. Then, convolutional filters are applied to the matrix of word vectors to extract information for text classification.

Like a sentence, which is an ordered sequence of words, a playlist is an ordered sequence of songs. This observation motivates us to apply CNN to playlist classification and to exploit the concept of embedding to represent each song as a fixed-dimension vector. Although audio signal features are the input to most automatic music classification models [7]–[9], [16], they in fact can be mapped into a song embedding space [17]. We focus on the use of song embedding as the basis of playlist classification because it can be done without the need of audio signal processing and because it allows the classifier to be trained without concerning the semantic gap between audio signals and labels [16]. Besides, applying convolutional filters to the matrix of song embedding vectors is more efficient and flexible than applying other neural network structures such as recurrent neural network [12].

III. Proposed Model

The proposed CNN model, as shown in Fig. 1, consists of an embedding layer, a convolutional layer, and a classification layer. The details of each layer are described in this section.

A. Embedding Layer

This layer projects a song (indicated by a song ID) into a continuous embedding space $w_e \in R^{n \times d}$, where n is the total number of songs and d is the dimension of the embedding. Each row of w_e is the vector for a song. The song vectors are initialized with random values and refined jointly with other model parameters during the training process.

Let $\{s_1, s_2, \dots, s_L\}$ be a sequence of songs in an input playlist, where L is the length of the playlist. Each song in the playlist is represented by a one-hot vector $s_l \in R^{1 \times n}$ (n is the total number of songs), which has value '1' for the element corresponding to the integer mapping of the song and value "0" for the other elements. Then, the song can be projected into the d-dimensional embedding space by multiplying s_l by the embedding matrix w_e,

$$p_l = s_l w_e \,. \tag{1}$$

Therefore, the input playlist can be represented as a matrix, where each row is the embedding vector of one song in the playlist. Because each playlist has its own number of songs, the playlist matrix is zero-padded to ensure every playlist matrix input to the convolutional layer has the same size.

B. Convolutional Layer

The playlist matrix is then input to the convolutional layer, which involves a sequence of 2-D and 1-D convolution operations [18]. To extract information of different granularity from the playlist matrix, our model is structured to have two parallel convolution paths, each with its own filter size for 2-D convolution. In each path, the playlist matrix is first convolved with a set of filters by 2-D convolution to generate feature maps. A rectified linear unit (ReLU) [19] is adopted as an activation function at the end of each convolution. Then the resulting feature maps are convolved with another set of filters by 1-D convolution. Like the 2-D convolution step, a ReLU function is applied at the end of each 1-D convolution. Finally, a max-pooling operation is performed for dimension reduction. At the end of the convolutional layer, all the features obtained in the two convolution paths are concatenated into a vector that serves as a global feature of the input playlist.

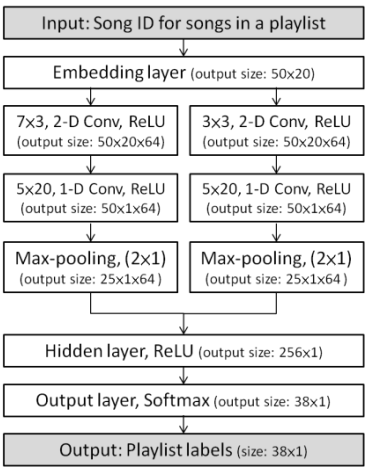

Fig. 1. Network structure of the proposed model

Although having two convolution operations in each path seems a shallow CNN, it is powerful enough to extract information from a playlist matrix because the convolutional filters are fine-tuned together with the song embedding vectors. Applying more convolution operations results in over-fitting problem.

C. Classification Layer

The global feature vector is then fed into the classification layer for mapping into an output vector $o \in R^{t \times 1}$, where t is the total number of labels. Note that it is a k-hot vector because multiple labels are allowed.

The classification layer is a multilayer perceptron consisting of a hidden layer and an output layer. Given an global feature vector $c \in R^{m \times 1}$, the hidden layer generates a vector $h \in R^{q \times 1}$ by the following ReLU activation function:

$$h = relu(w_h c + b_h) \,, \tag{2}$$

where $w_h \in R^{q \times m}$ and $b_h \in R^{q \times 1}$ are the weight matrix and the bias vector, respectively, of the hidden layer. Finally, the output layer adopts a sigmoid activation function for multi-label classification [20], which generates the output vector $o \in R^{t \times 1}$ by

$$o = sigmoid(w_o h + b_o) \,, \tag{3}$$

where $w_o \in R^{t \times q}$ and $b_o \in R^{t \times 1}$ are the weight matrix and the bias vector, respectively, of the output layer.

The trainable parameters in our model include the song embedding matrix, the convolutional filters, and the weight matrices and the bias vectors of the classification layer. These

parameters are fine-tuned jointly by optimizing the loss function L, which is computed by summing up the binary cross-entropy for every corresponding element between the model output o^i and the target label y^i for every training sample i:

$$L = -\sum_{i}^{N}\sum_{j}^{t}\left[y_j^i \, log\left(o_j^i\right) + \left(1 - y_j^i\right)\log\left(1 - o_j^i\right)\right], \quad (4)$$

where N is the total number of training samples, t is the total number of labels, and y_j^i is '1' when training sample i has label j, otherwise, it is '0'.

IV. Data Augmentation Techniques

Because a playlist can comprise any songs, the training samples should have good diversity; otherwise, a playlist classification model suffers from over-fitting easily. We propose two playlist data augmentation techniques to prevent our model from over-fitting.

A. Playlist Sub-Sampling

This technique generates a new playlist by randomly taking a subset of songs from an existing playlist. Like the situation where a user removes a number of songs from a playlist, the theme of the playlist is unlikely to change. By increasing the number and the diversity of training samples, this technique can improve the robustness of our model. It also allows us to analyze the effect of song order on playlist classification, because the song order of playlists may be disrupted after sub-sampling.

B. Song Dropout

Similar to the word dropout for natural language sentences [21], this technique randomly replaces a number of songs in a playlist with a symbol "UNK" (an abbreviation of unknown). The symbol "UNK" is then treated in the same way as other songs and represented by a vector in the embedding layer. This technique deals with the existence of some songs in a playlist which are unrelated to the theme of the playlist and thereby prevents the interference of such noise songs to the stability of our model.

V. Experimental Setup

This section describes the details of our experiment, including our playlist dataset, the baseline models for comparison, the parameter setup of the proposed CNN model, and the evaluation metrics for playlist classification.

A. Playlist Dataset

The dataset applied in this work contains a total of 11,445 playlists created by the music experts at KKBOX[1], which is a leading music streaming service provider in Asia. Each playlist is assigned by the experts multiple labels, such as language-related labels (e.g., western, mandarin, and japanese), genre-related labels (e.g., rock, jazz, and electronic), event-related labels (e.g., fitness, party, and sleeping) or mood-related labels (e.g., love, happy, and sad). In average, each playlist has 20 songs and two labels.

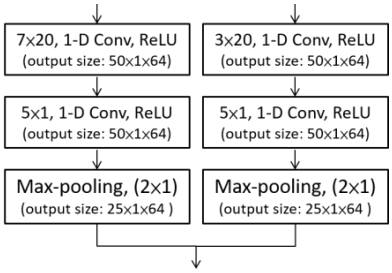

Fig. 2. The convolutional layer of the basic CNN model

To ensure that each label has a sufficient number of playlists for training, only the top 38 frequent labels were considered as the classification targets in the experiment. Note that the distribution of the labels is not balanced; the most frequent label is used 1,965 times (meaning found in 1,965 playlists) while the 38th frequent label is used only 97 times. We then filtered out the playlists that do not have any of the top 38 labels, leaving a total of 9,410 playlists and 102,213 songs for the experiment. We randomly split the remaining playlists into testing set and training set: 10% of the playlists to be the testing set and the other 90% to be the training set.

Note that the two data augmentation techniques described in Sec. IV were applied to only the playlists in the training set. Specifically, the playlist sub-sampling technique extracted ten new playlists from each playlist in the training data, and the song dropout technique replaced songs by "UNK" symbol with the rate 0.2, meaning that two out of ten songs in a playlist were replaced.

B. Models for Comparison

For the comparison, we implemented two classification models using support vector machine (SVM) [22] and *k*-nearest neighbors (*k*NN) [23], respectively. The input of the two models is the k-hot vector of a playlist, where an element of the vector is '1' if the corresponding song exists in the playlist, otherwise, it is '0'. For the *k*NN model, only the nearest neighbor is used for classification in our experiment. We considered these two basic classifiers as our baseline models because there was no previous model proposed for playlist classification.

Besides, we constructed another CNN model to show the effectiveness of applying 2-D convolutional filters in the proposed model. The CNN model has an identical network structure of the proposed model except that it uses only 1-D convolutional filters in the convolution layer to extract information from a playlist matrix, as shown in Fig. 2. We indicate the CNN model as the basic CNN model in the following sections.

[1] KKBOX: https://www.kkbox.com/

TABLE I. PROPOSED MODEL VS. BASIC CNN MODEL WITH OR WITHOUT SONG DROPOUT AND PLAYLIST SUB-SAMPLING

Model	Song dropout	Playlist sub-sampling	Precision	Recall	F_1 score
Basic CNN	-	-	0.3660	0.2252	0.2787
Basic CNN	V	-	0.3669	0.2079	0.2654
Basic CNN	-	V	0.4811	0.3099	0.3769
Basic CNN	V	V	0.5238	0.3243	0.4004
Proposed	-	-	0.3225	0.2752	0.2969
Proposed	V	-	0.3341	0.2276	0.2707
Proposed	-	V	0.4528	0.4676	0.4604
Proposed	V	V	**0.5585**	**0.4968**	**0.5257**

C. Parameter Setup

The details of parameter setup of the proposed model are described as the following: The dimension of the embedding space was 20, and every playlist matrix was zero-padded to (50×20). The number of filters for all convolution steps was 64. The size of the 2-D filters was (3×3) and (7×3) in the two parallel convolution paths, respectively, and the size of 1-D filters was both (5×20). The max-pooling size was (2×1). The output dimension of the hidden layer was 256, and the output dimension of the output layer was 38, which was the number of the target labels. The training was done using RMSprop optimizer [24] with a learning rate of 1e-4 and a batch size of 128.

For the basic CNN model, the size of the 1-D filters in the first convolution step was (3×20) and (7×20) in the two parallel convolution paths, respectively, and the size of the 1-D filters in the second convolution step was both (5×1). The other parameter setup of the basic CNN model was the same as the proposed model.

D. Evaluation Metrics

Two standard metrics, precision and recall, were applied for the measurement of classification performance:

$$precision = \frac{|L_P \cap L_G|}{|L_P|}, \quad (5)$$

$$recall = \frac{|L_P \cap L_G|}{|L_G|}, \quad (6)$$

where L_P is the set of the labels that are predicted by a model to be related to a playlist, and L_G is the set of ground truth labels that are truly related to the playlist. A high precision means that most predicted labels are related to the playlist, and a high recall means that most related labels are predicted by the model. Besides, F_1 score, the harmonic average of precision and recall, was calculated by

$$F_1 \text{ score} = 2 \cdot \frac{precision \cdot recall}{precision + recall}. \quad (7)$$

A high F_1 score means the model has a high precision and a high recall at the same time. We computed the three metrics (precision, recall, and F_1 score) for all testing playlists and averaged the results to measure the performance of a classification model.

TABLE II. PROPOSED MODEL VS. OTHER MODELS

Model	Precision	Recall	F_1 score
*k*NN	0.3408	0.3819	0.3450
SVM	0.4024	0.3679	0.3699
Basic CNN	0.5238	0.3243	0.4004
Proposed	**0.5585**	**0.4968**	**0.5257**

VI. RESULTS AND DISCUSSIONS

In this section, we first describe the comparison between the proposed model and the basic CNN model and show the effectiveness of the proposed playlist sub-sampling and song dropout techniques. Then, we make an overall performance comparison of the proposed model, the basic CNN model, the SVM model, and the *k*NN model. Finally, we further examine the labels predicted by the proposed model and compare them with the ground truth labels.

A. Proposed Model vs. Basic CNN Model

Table I shows the evaluation results of the proposed model and the basic CNN model with or without the two playlist data augmentation techniques. We can first see that when only the song dropout technique was applied, a lower F_1 score for both models was resulted. It is because that the song dropout technique may introduce out-of-vocabulary (OOV) problem, which means that some songs in a testing playlist are absent from the training data. This is further confirmed by noting that the F_1 score of both models was improved, when the playlist sub-sampling technique was applied ahead of the song dropout technique. The playlist sub-sampling technique increases the number of times that each song appears in the training data and lowers the incidence of OOV. Second, we can see that although the playlist sub-sampling technique disrupted the song order of the playlists for training, it significantly improved the performance of both models. It suggests that the song order of playlists has no practical effect for training the CNN-based playlist classification models.

Finally, we can see that the proposed model had a higher F_1 score than the basic CNN model in all cases. The difference between the two models was more apparent when the playlist sub-sampling technique was applied. Note that 2-D convolution was applied to a playlist matrix along both horizontal and vertical directions (two dimensions), and 1-D

TABLE III. LABELLING RESULT

Ground truth labels	"love", "summer", "electronic/dance"
Predicted labels	"western", "party ", "electronic/dance"

Song name	**Artist**
Sweet Summer Sins	Juloboy feat. Mougleta
Hunter	Galantis
Attention	CHARLIE PUTH
On Your Side	The Veronicas
Sweet Addiction	Yuksek
Shape of You	Ed Sheeran
One More Weekend	Loui & Scibi feat. Nuwella
Hot2Touch	Felix Jaehn, Hight, Alex Aiono
Be My Love	Mahalo feat. Cat Lewis
On My Mind	3LAU
First Time	Kygo, Ellie Goulding
My Love	Wale
In Your Arms	NERVO
Summer Love	Various Artists
Nights With You	MØ
Trust Nobody	Cashmere Cat
No Vacancy	OneRepublic
Every Little Thing	Deepend
Feels	Calvin Harris
Never Wanna Lose You	CLMD
Girls Like Girls	Hayley Kiyoko
Bad Girl	Isle of Skye
Love Song	Late Night Alumni
Feel Good	Gryffin, Illenium
Call You Mine	Douchka
Love Somebody	Justin Caruso
Feel My Love	Trivecta

convolution was only applied along vertical direction. Our results verify that 2-D convolutional filters can indeed extract information from a playlist matrix more effectively than 1-D convolutional filters.

B. *Proposed Model vs. Other Models*

Table II shows the overall performance comparison of the proposed model, basic CNN model, SVM model, and *k*NN model. For fair comparison, the playlist sub-sampling technique was also applied to the *k*NN and SVM models, while the song dropout technique cannot be applied because the two models do not have song embedding.

The results show that both the proposed model and basic CNN model performed significantly better than the *k*NN and SVM models. For the *k*NN model, we found that the best F_1 score was achieved when the parameter k was set to 1. This indicates that the playlists hardly have a large number of identical songs, so the *k*NN model cannot find more than one playlist to improve the performance. Likewise, the SVM model cannot find a good optimal support vector for classification, because the data points were extremely sparse in the feature space of the SVM model. In contrast, the CNN-based models can avoid this sparsity problem by learning dense song embedding vectors and extracting information from a playlist matrix directly.

TABLE IV. LABELLING RESULT

Ground truth labels	"happy"
Predicted labels	"excitation"

Song name	**Artist**
The Greatest	Sia
Closer	The Chainsmokers
Chen 52	Fire EX.
Faded	Alan Walker
My Way	Calvin Harris
Love Yourself	Justin Bieber
Departure	Christine Fan
Irreplaceable	S.H.E
Tough	Mayday
Don't Let Me Down	The Chainsmokers
Round And Around	Erika
Princess	Jam Hsiao
We Don't Talk Anymore	CHARLIE PUTH
Watch Me Do	Meghan Trainor
If We Meet Again	William Wei

C. *Model-Predicted Labels*

We further show the effectiveness of our model by examining the predicted labels for testing playlists. Two examples are shown in Tables III and IV, where the labels predicted by the proposed model and the ground truth labels assigned by labelers are listed together. We found that the labelers (music experts) may focus on only the theme they are familiar with while neglect other possible labels. The playlist shown in table III consists of Western songs with a strong rhythm suitable for playing in a dance party. The ground truth labels are "love", "summer", and "electronic/dance," while the labels "western" and "party" predicted by the proposed model also match this playlist quite well. The other playlist shown in Table IV has the ground truth label "happy", and the predicted label is "excitation". Although the two labels may seem different, the fact is that the lyrics of the songs in the playlist are full of encouragement and have positive meaning. Therefore, the label "excitation" may also be good for the playlist. The results clearly illustrate that the proposed model can provide equally reasonable labels for playlists and enrich the playlist labeling.

VII. CONCLUSIONS

We have described a novel multi-label model for playlist classification based on a convolutional neural network. Our model automatically learns a song embedding space and avoids the need for extracting features from audio signals. The two data augmentation techniques, playlist sub-sampling and song dropout, further help the model to prevent from over-fitting and to enhance the classification performance. The experimental results demonstrated that the song order of playlists has no effect on playlist classification. This is particularly important from the practical point of view because users may randomly shuffle songs when playing a playlist. The proposed model performs significantly better than *k*NN and SVM models and can assist users and service providers to enrich playlist labels.

REFERENCES

[1] T. Leong, S. Howard, and F. Vetere, "Choice: Abdicating or exercising?" in *Proc. CHI Conf. Human Factors Computing*, pp. 715–724, 2008.

[2] M. Schedl, H. Zamani, C.-W. Chen, Y. Deldjoo, and M. Elahi, "Recsys challenge 2018: Automatic playlist continuation," in *Late-Breaking/Demos Proc. 18th Int. Soc. Music Information Retrieval Conf.*, 2017.

[3] S. Cunningham, D. Bainbridge, and A. Falconer, "More of an art than a science: Supporting the creation of playlists and mixes," in *Proc. Int. Symposium Music Information Retrieval*, pp. 240–245, 2006.

[4] B. Fields, "Contextualize your listening: The playlist as recommendation engine," Ph.D. thesis, University of London, 2011.

[5] D. Jannach, I. Kamehkhosh, and G. Bonnin, "Analyzing the characteristics of shared playlists for music recommendation," in *Proc. Recommender Systems Social Web*, 2014.

[6] G. Bonnin and D. Jannach, "Automated generation of music playlists: Survey and experiments," *ACM Computing Surveys*, vol. 47, no. 2, pp. 26, 2014.

[7] S. Dieleman and B. Schrauwen, "End-to-end learning for music audio," in *Proc. Int. Conf. Acoustics Speech Signal Processing*, pp. 6964–6968, 2014.

[8] K. Choi, G. Fazekas, and M. Sandler, "Automatic tagging using deep convolutional neural networks," in *Proc. Int. Symposium Music Information Retrieval*, 2016.

[9] K. Choi, G. Fazekas, M. Sandler, and K. Cho, "Convolutional recurrent neural networks for music classification," in *Proc. Int. Conf. Acoustics Speech Signal Processing*, 2017

[10] J. Bissmark and O. Wärnling, "The sparse data problem within classification algorithms : The effect of sparse data on the naïve Bayes algorithm," B.S. thesis, KTH Royal Institute of Technology, 2017.

[11] Y. Kim, "Convolutional neural networks for sentence classification," in *Proc. Empirical Methods Natural Language Processing*, pp. 1746–1751, 2014.

[12] C. N. dos Santos and M. Gatti, "Deep convolutional neural networks for sentiment analysis of short texts," in *Proc. Computational Linguistics*, pp. 69–78, 2014.

[13] W. Yin and H. Schutze, "Multichannel variable-size convolution for sentence classification," in *Proc. Computational Natural Language Learning*, pp. 204–214, 2015.

[14] X. Yang, C. Macdonald, and I. Ounis, "Using word embeddings in twitter election classification," *arXiv preprint arXiv:1606.07006*, 2016.

[15] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv: 1301.3781*, 2013.

[16] Z. Fu, G. Lu, K.-M. Ting, and D. Zhang, "A survey of audio-based music classification and annotation," *IEEE Trans. Multimedia*, vol. 13, no. 2, pp. 303–319, 2011.

[17] A. Van de Oord, S. Dieleman, and B. Schrauwen, "Deep content-based music recommendation," in *Proc. Advances Neural Information Processing Systems*, pp. 2643–2651, 2013.

[18] O. Abdel-Hamid, A.-R. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, "Convolutional neural networks for speech recognition," *IEEE/ACM Trans. Audio Speech Language Processing*, vol. 22, no. 10, pp. 1533–1545, 2014.

[19] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. 27th Int. Conf. Machine Learning*, pp. 807–814, 2010.

[20] J. Nam, J. Kim, E. L. Mencía, I. Gurevych, and J. Fürnkranz, "Large-scale multi-label text classification—Revisiting neural networks," in *Proc. Joint European Conf. Machine Learning Knowledge Discovery Databases*, pp. 437–452, 2014.

[21] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio, "Generating sentences from a continuous space," *arXiv preprint arXiv:1511.06349*, 2015.

[22] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics Computing*, vol. 14, no. 3, pp. 199–222, 2004.

[23] S. Tan, "An effective refinement strategy for KNN text classifier," *Expert Systems Applications*, vol. 30, no. 2, pp. 290–298, 2006.

[24] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.