Neural semi-supervised learning for multi-labeled short-texts

Johnny Torres and Carmen Vaca

ESPOL Polytechnic University
Department of Electrical and Computer Engineering (FIEC)
jomatorr,cvaca@espol.edu.ec

Abstract. The massive data generated by users in online platforms, such as social networks, create challenges for text classification tasks based on supervised learning. Supervised learning often requires a lot of feature engineering or a significant amount of annotated data to achieve good results. However, the scarcity of annotated data is a critical issue, and manual annotation can be both costly and time-consuming. Semi-supervised learning requires far less annotated data and achieve similar performance as supervised approaches. In this paper, we introduce a semi-supervised neural architecture for muti-label settings, that combines deep learning representation and *k-means* clustering. The results show that the semi-supervised approach can leverage large-scale unlabeled data and achieve better results compared to baseline unsupervised as well as supervised methods.

1 Introduction

The classification or grouping of short texts is critical in various tasks in text mining and the retrieval of information in the context of social networks or data generated by users on the web. Specifically, these tasks aim to categorize or group similar texts, so that texts with the same label or group are similar to each other and different from texts in other categories or groups. Traditional classification or grouping models often use a sparse representation for text data, such as the bag of words (BOW) or TF-IDF [12].

However, the characteristics of the short texts create some problems for both conventional unsupervised and supervised models. Usually, the number of unique words in each short text is small (90% of the texts instances in the HappyDB dataset have less than 23 words), and as a result, the problem of lexical shortage generally leads to poor grouping quality [8].

An alternative to address lexical shortages is to enrich text representations by extracting characteristics and relationships with sources such as Wikipedia [4] or ontologies [9]; however, this approach requires written knowledge, which also depends on the language. Other alternative is to code texts in distributed dense vectors [14] with neural networks [18].

Another problem is the definition of the labels for a specific task and the number of manually annotated instances for each label. Unsupervised methods learn the categories from the data, but the resulting groupings may not be related to the expected labels. Supervised methods have predefined labels but often require a considerable number of labeled instances to learn to categorize. Semi-supervised approaches offer an alternative

to solve these problems by using a small amount of labeled data according to predefined classes, at the same time take advantage of the massive unlabelled data availability [3].

This paper investigates the research question: How can a semi-supervised approach learn to categorize short texts in a multi-label taxonomy using a small set of labeled data and leveraging the availability of large amounts of unlabeled data? To that end, we build upon neural semi-supervised *k-means* clustering that modifies the conventional objective function and adds a penalty term for labeled data [17]. We extended the neural semi-supervised clustering and applied to multi-label settings. The results show that semi-supervised *k-means* outperform other baseline unsupervised models for multi-label classification tasks.

The rest of the paper is structured as follows: a) we review related work and the *k-means* clustering, b) we describe the neural semi-supervised clustering for multi-label setting, c) we analysis the experimental results, and d) finally, we outline the conclusions and future work.

2 Related Work

Previous work on semi-supervised clustering methods analyzes methods based on: *constraints* and *representation*. The *constraint-based approaches* use a small percentage of labeled data to restrict the clustering process [7]. Instead, the *representation-based methods* first learn a data representation model that satisfies the labeled data, and then use it to group both labeled and unlabeled data [3].

The hybrid approaches try to integrate both methods in a unified framework [5]; However, the use of linear projection for learning by representation has limitations to achieve good performance. Recent methods use deep neural architectures to learn text representations that overcome the limitation of linear models [18]. However, the separation of the learning process of the data representation model and the clustering model restrict the benefits and is more similar to the techniques *representation-based*. In this work, the proposed model builds on an approach that combines into an integrated framework both the representation of deep learning and the clustering method [17].

3 Unsupervised Learning

In unsupervised learning, *k-means* is an algorithm for clustering data used in many applications, including text mining tasks [6]. The *k-means* algorithm divides the data into a K number of clusters, so that minimizes the distance of each point to the centroids of the clusters, assigning it to the nearest cluster. The input to the clustering model are the set of short texts $\{s_1, s_2, s_3, ..., s_N\}$ represented by the data points $\{x_1, x_2, x_3, ..., x_N\}$, where x_i is a sparse or dense vector.

The *k-means* algorithm defines a set of binary variables $r_{nk} \in \{0, 1\}$ for each data point x_n, where $k \in \{1, ..., K\}$ specifies the cluster assigned. For example, $r_{nk} = 1$ if x_n is assigned to cluster k, and $r_{nj} = 0$ for $j \neq k$. The objective function in *k-means* is defined as:

$$J_{unsup} = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \|x_n - \mu_k\|^2 \tag{1}$$

where μ_k is the centroid of the k-th cluster. The *k-means* algorithm learns the values of $\{r_{nk}\}$ and $\{\mu_k\}$ such that optimizes J_{unsup}. To minimize the objective function, *k-means* utilizes the gradient descent approach with an iterative procedure [16].

Each iteration involves two steps: *assign clusters* and *estimate centroids*. In the *assign clusters* step, *k-means* minimizes J_{unsup} with respect to $\{r_{nk}\}$ by keeping fixed $\{\mu_k\}$. In this case, J_{unsup} is a linear function for $\{r_{nk}\}$, so that we optimize each data point separately by merely assigning the n-th data point to the closest cluster centroid.

In the *estimate centroids* step, *k-means* minimizes J_{unsup} with respect to $\{\mu_k\}$ by keeping $\{r_{nk}\}$ fixed. In this case, J_{unsup} is quadratic function of $\{\mu_k\}$, and we minimize it by setting to zero the derivative for $\{\mu_k\}$, as follows:

$$\frac{\partial J_{unsup}}{\partial \mu_k} = 2 \sum_{n=1}^{N} r_{nk}(x_n - \mu_k) = 0 \tag{2}$$

Then, we can solve $\{\mu_k\}$ as

$$\mu_k = \frac{\sum_{n=1}^{N} r_{nk} x_n}{\sum_{n=1}^{N} r_{nk}} \tag{3}$$

Thus, μ_k corresponds to the mean of all the data points assigned to the cluster k.

4 Neural Semi-supervised Clustering

The classical *k-means* algorithm uses unlabeled data to solve the clustering problem based on an unsupervised learning approach; however, the clustering results may not be consistent with the expected labels. We extend the semi-supervised approach in [17], which injects some supervised information into the learning process to produce useful and coherent clusters. Similar to the classic *k-means* algorithm, the training steps for the neural semi-supervised *k-means* are:

1. Initialize $\{\mu_k\}$ and $f(\cdot)$.
2. Repeat until convergence:
   (a) *assign clusters*: Assigns each short-text to its nearest cluster centroid based on its neural representation.
   (b) *estimate centroids*: Estimates the clusters' centroid based on the cluster assignments from previous step.
   (c) *update parameters*: Updates the neural networks parameters according to the objective function by keeping fixed the centroids and cluster assignments.

4.1 Representation Learning

We represent each short text entry s_i as a sequence of word indices and together with the initial centroids form the input data to the semi-supervised neuronal clustering model. Then, the embedding layer maps each word in the sequence as a dense vector $x = f(s)$, using word embeddings initialized randomly or from pre-trained embeddings [14, 13]. In this approach, rather than training the text representation model independently, the semi-supervised clustering integrates it with the *k-means* algorithm training process.

4.2 Objective Function

The neural semi-supervised clustering uses a small number of labeled instances to guide the clustering process and minimizes the objective function defined as:

$$\begin{aligned} J_{semi} = \sum_{c=1}^{C}\{\alpha \sum_{n=1}^{N}\sum_{k=1}^{K} r_{nk}\|f(s_n) - \mu_k\|^2 \\ + (1-\alpha)\sum_{n=1}^{L}\{\|f(s_n) - \mu_{g_n}\|^2 + \\ \sum_{j \neq g_n}[l + \|f(s_n) - \mu_{g_n}\|^2 - \|f(s_n) - \mu_j\|^2]_+\}\} \end{aligned} \tag{4}$$

where $\{(s_1, y_1), (s_2, y_2), ..., (s_L, y_L)\}$ denote the labeled data, and the unlabeled data is $\{s_{L+1}, s_{L+2}, ..., s_N\}$. The label y_i specify the cluster for each short-text s_i. The outer sum iterates over the number of labels C defined in the taxonomy; thus, extending the original objective function in [17]. The objective function contains two terms:

1. The first term is the objective function in the classic *k-means* algorithm (Eq. 1), and the second term penalizes depending how far are the predicted clusters from the ground-truth clusters for labeled data. The factor $\alpha \in [0, 1]$ is used to tune the importance of unlabeled data.
2. The second term contains two sub-terms:
   (a) The first sub-term penalizes depending on the distance between each labeled instance and its correct cluster centroid, where $g_n = G(y_n)$ indicates the cluster ID given by the label y_n. The mapping function $G(\cdot)$ uses the Hungarian algorithm [15].
   (b) The second sub-term specifies a hinge loss function with a margin l, where $[x]_+ = max(x, 0)$. This term incurs in some loss if the distance to the ground truth centroid is larger (by a margin l) than the distances to the wrong centroids.

4.3 Model Training

The parameters in J_{semi} are: the clusters' assignment for each text $\{r_{nk}\}$, the clusters' centroids $\{\mu_k\}$, and the neural network weights $f(\cdot)$. The goal is to find the values of

$\{r_{nk}\}$, $\{\mu_k\}$, and parameters in $f(\cdot)$ that minimizes J_{semi}. Based on the *k-means* algorithm, the semi-supervised model iteratively minimizes J_{semi} with respect to $\{r_{nk}\}$, $\{\mu_k\}$, and parameters in $f(\cdot)$.

First, the model *initializes the clusters' centroids* $\{\mu_k\}$ with the *k-means* method [1], and also initializes randomly the parameters in the neural network. Then, the model iteratively carry out three steps (*assigns clusters*, *estimates the centroids*, and *updates the parameters*) until J_{semi} converges.

The *assign clusters* step minimizes J_{semi} with respect to $\{r_{nk}\}$ by keeping fixed $f(\cdot)$ and $\{\mu_k\}$ to assign a cluster ID for each data point. The second term in Eq. (4) has no relation with $\{r_{nk}\}$. Thus, the model only needs to minimize the first term, by setting the nearest cluster centroid to each text, i.e., is identical to the *assign clusters* step in the *k-means* algorithm. In this step, the model also calculates the mappings between the ground-truth clusters specified by $\{y_i\}$ and the cluster assignments for the labeled data.

The *estimate centroids* step minimizes J_{semi} with respect to $\{\mu_k\}$ by keeping $\{r_{nk}\}$ and $f(\cdot)$ fixed, which corresponds to the *estimate centroids* step in the classic *k-means* algorithm. It aims to estimate the cluster centroids $\{\mu_k\}$ based on the cluster assignments $\{r_{nk}\}$ from the *assign_cluster* step. In the Eq. 4, the second term considers each labeled instance in the process of estimating cluster centroids. By solving $\partial J_{semi}/\partial \mu_k = 0$, we get

$$\mu_k = \frac{\sum_{n=1}^{N} \alpha r_{nk} f(s_n) + \sum_{n=1}^{L} w_{nk} f(s_n)}{\sum_{n=1}^{N} \alpha r_{nk} + \sum_{n=1}^{L} w_{nk}} \tag{5}$$

$$\begin{aligned}
w_{nk} &= (1-\alpha)(I^{'}_{nk} + \sum_{j \neq g_n} I^{''}_{nkj} - \sum_{j \neq g_n} I^{'''}_{nkj}) \\
I^{'}_{nk} &= \delta(k, g_n) \\
I^{''}_{nkj} &= \delta(k, j) \cdot \delta^{'}_{nj} \\
I^{'''}_{nkj} &= (1 - \delta(k, j)) \cdot \delta^{'}_{nj} \\
\delta^{'}_{nj} &= \delta(l + \|f(s_n) - \mu_{g_n}\|^2 - \|f(s_n) - \mu_j\|^2 > 0)
\end{aligned} \tag{6}$$

where $\delta(x_1, x_2)$=1 if x_1 is equal to x_2, otherwise $\delta(x_1, x_2)$=0; and $\delta(x) = 1$ if x is true, otherwise $\delta(x) = 0$. In the numerator of Eq. 5, the first term represents the contributions from all data points, and the weight of s_n for μ_k is αr_{nk}. The second term represents labeled data, and w_{nk} is the weight of an instance s_n for μ_k.

The *update parameters* step minimizes J_{semi} with respect to $f(\cdot)$ by keeping $\{r_{nk}\}$ and $\{\mu_k\}$ fixed, with no counterpart in the *k-means* algorithm. The primary goal is to learn the parameters of the text representation model. The training uses J_{semi} as the loss function and employs *Adam* algorithm to optimize it [11].

5 Experiments

5.1 Experimental Setting

We evaluate the models on the HappyDB dataset [2], comprising individual accounts of *happy moments*. The aim is to predict *agency* and *social* labels that indicate the context

of happy moments. For training, we use a small subset of labeled data and a large subset unlabeled dataset [10]. The table 1 summarizes the number of labeled and unlabeled text instances for training, as well as the number of text instances in the test set. For the experiments, the splitting strategy is to randomly sample 80% of labeled instances for training (*training set*) and remaining 20% instances for validation (*validation set*). The unsupervised and semisupervised models use the unlabeled instances for training (*unlabeled set*). We train the models using k-fold cross-validation ($k = 10$) on the training set and report the results for the validation set using the metric F_1.

Table 1. Statistics for the HappyDB dataset

Dataset	Labeled	Unlabeled	Test	Total
HappyDB	10,560	72,324	17,215	100,099

5.2 Model Hyperparameters

Neural architectures introduce several *hyper-parameters* like the output dimension of the text representation models, while semi-supervised *k-means* clustering has α in Eq. 4. This subsection analyzes the impact of some of the hyper-parameters and determines the configuration for further experiments.

Embeddings dimension To evaluate the effectiveness of the output dimension in text representation models, we perform experiments with embeddings size of {50, 100, 200, 300, 500, 1000}, while maintaining the all other parameters fixed. Figure 1 show that the score F_1 drops if the size is ≤ 100 and the curve falls if the size is ≥ 500. Based on the results, we use 300 as the embeddings size.

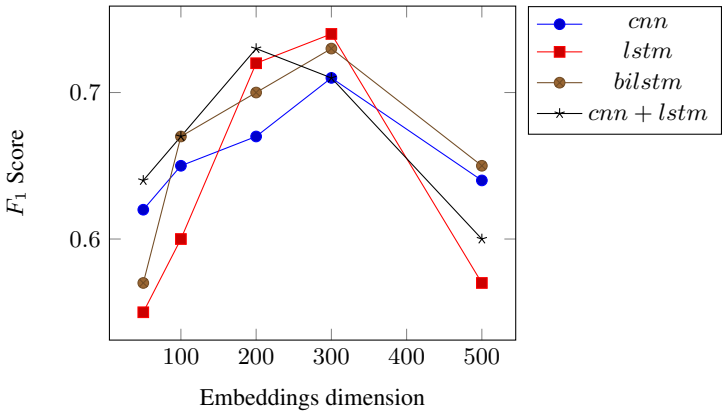

Fig. 1. Influence of the dimensionality of the text learning representation.

Alpha We evaluate the effect of α in Eq. 4, which indicates the importance of unlabeled data in the performance of the model. We test α with values of: $\{0.00001, 0.0001, 0.001, 0.01, 0.1\}$, and maintain the other parameters fixed. The figure 2 shows that the performance decay for small α values. By increasing the value of α, we notice progressive improvements and reach a peak F_1 score with $\alpha = 0.1$. Further experiments use the value of $\alpha = 0.1$ as it maximizes F_1.

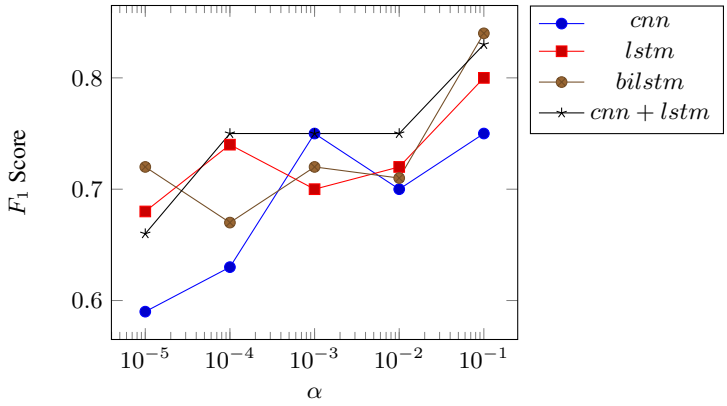

Fig. 2. Influence of unlabeled data, where the x-axis is α in Eq. (4).

Labeled set size This parameter controls the influence of the size of the labeled data. We evaluate the ratio of labeled data for training between $[1\%, 10\%]$, and kept the other parameters fixed. The figure 3 illustrates the performance improvement as the size of labeled data increases and confirms the importance of labeled data for training.

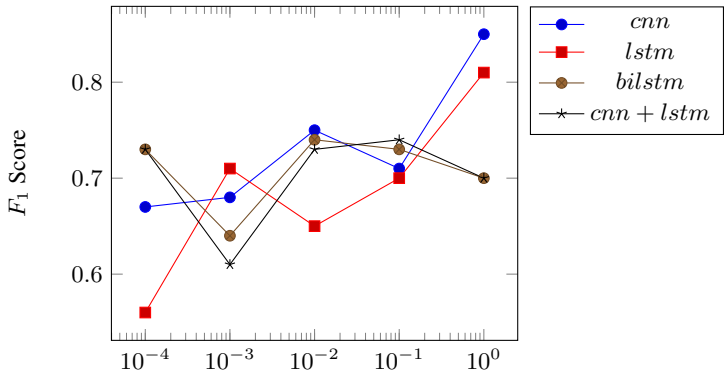

Fig. 3. Influence of the size of labeled data used for training.

Pre-training This option measures the effect of the pre-training embeddings for neural architectures. We use pre-trained embeddings in the models as a classification task with labeled data, and we then use the weights (excluding the top layer) to initialize semi-supervised clustering. We evaluate several pre-trained embeddings such as Word2Vec, Glove, FastText. Figure 4 shows that pre-trained embeddings achieve superior performance compared to random embeddings; for further experiments we use FastText.

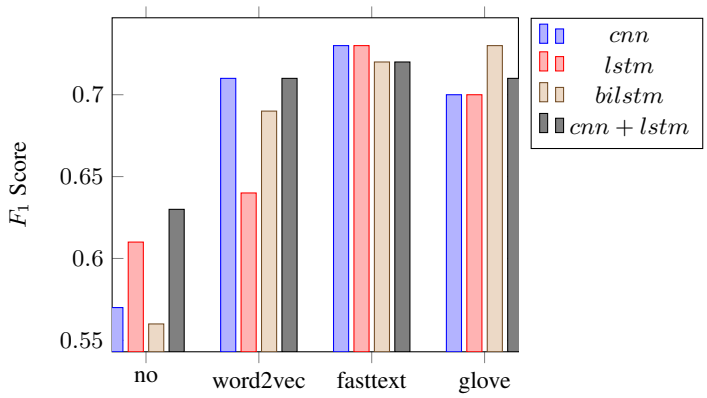

Fig. 4. Influence using pre-training embeddings in neural models.

5.3 Models Benchmarking

This subsection compares the proposed semi-supervised approach with unsupervised and supervised models.

Unsupervised learning: All unsupervised models use *k-means* for clustering. We cluster with $k = 2$ to map the values for each label $(0, 1)$. For learning representation, we use the following methods:

- BOW: represents each short-text as sparse vector based on term frequency (TF).
- TF-IDF: similar to BOW, uses a sparse vector to represent each short-text based on term frequency-inverse document frequency.
- AVG-EMB: uses word embeddings vectors to represent each short-text and then calculate the average.

Supervised learning: We evaluate several supervised models for the classification task, the representation learning used depends on each model as described next:

- LR: uses sparse vector representation that feeds a logistic regression classifier.
- FastText: uses dense word vectors representation (embeddings layer), followed by a Global Average Pooling layer, which averages the word embeddings, and then uses a Dense layer with sigmoid activation to predict the labels.

- CNN: uses dense word vectors representation (word embeddings layer) followed by a Dropout layer, then a convolutional layer, and an output layer with sigmoid activation.
- LSTM: similar to CNN, but the word embeddings layer feeds a recurrent LSTM layer, which is more suitable for sequence modeling such as texts.
- BiLSTM: uses two LSTM networks to model the texts sequences in both directions, followed by a Dropout layer with rate 0.5, and then a dense layer with sigmoid activation.
- CNN-LSTM: leverage the advantage of CNN layer to capture salient features and sequence modeling capability of LSTM.

Table 2 summarizes the scores of the models on the test set. The models fall into three categories (type): unsupervised, supervised, and semi-supervised. The metrics are precision, recall, and F_1. We report the scores for each label: agency and social. The last three columns show the total weighted score of the metrics for each model. The results show that the supervised systems outperform unsupervised models by a large margin, which highlights the importance of labeled data. Among the supervised learning, deep neural models perform better than the baseline method (LR); however only with a small margin. Finally, the semi-supervised model shows promising results, as it achieves score close to the supervised models.

Table 2. Performance of the models.

		Agency			**Social**			**Total (Weighted)**		
Type	**Model**	**Prec.**	**Recall**	F_1	**Prec.**	**Recall**	F_1	**Prec.**	**Recall**	F_1
unsupervised	BOW	72.77	38.51	50.37	54.46	74.32	62.86	63.61	56.42	56.61
	TF-IDF	72.70	84.69	78.24	54.02	52.96	53.49	63.36	68.83	65.86
	EMB-AVG	73.73	97.79	84.07	53.59	54.16	53.87	63.66	75.97	68.97
supervised	LR	81.06	94.30	87.18	93.00	82.30	87.32	87.03	88.30	87.25
	FastText	85.74	93.98	89.67	90.99	86.73	88.81	88.37	90.35	89.24
	CNN	90.92	88.53	89.71	89.34	90.53	89.93	90.13	89.53	89.82
	LSTM	89.19	89.81	89.50	91.57	87.52	89.50	90.38	88.67	89.50
	BiLSTM	89.18	90.84	90.00	92.84	84.96	88.72	91.01	87.90	89.36
	CNN-LSTM	89.38	88.92	89.15	89.70	87.88	88.78	89.54	88.40	88.96
semi-supervised	CNN	89.10	92.18	90.62	89.56	89.56	89.56	89.33	90.87	90.09

6 Conclusions

This work builds on the neural semi-supervised clustering that integrates a neural representation learning for short-texts, and the *k-means* clustering into a unified framework. To that end, the model utilizes a small percentage of labeled data to guide the intention for clustering. We extended the model to use it in muti-labeled clustering of short-texts. The results show that the proposed neural semi-supervised clustering is more effective than baselines unsupervised, and it close to the supervised models. Therefore, the results show the potential to overcome critical issues, such as of scarcity of labeled data, and leverage the availability of massive unlabeled data.

References

1. Arthur, D., Vassilvitskii, S.: k-means++: The advantages of careful seeding. In: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms. pp. 1027–1035. Society for Industrial and Applied Mathematics (2007)
2. Asai, A., Evensen, S., Golshan, B., Halevy, A., Li, V., Lopatenko, A., Stepanov, D., Suhara, Y., Tan, W.C., Xu, Y.: Happydb: A corpus of 100,000 crowdsourced happy moments. In: Proceedings of LREC 2018. European Language Resources Association (ELRA), Miyazaki, Japan (May 2018)
3. Bair, E.: Semi-supervised clustering methods. Wiley Interdisciplinary Reviews: Computational Statistics **5**(5), 349–361 (2013)
4. Banerjee, S., Ramanathan, K., Gupta, A.: Clustering short texts using wikipedia. In: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval. pp. 787–788. ACM (2007)
5. Bilenko, M., Basu, S., Mooney, R.J.: Integrating constraints and metric learning in semi-supervised clustering. In: Proceedings of the twenty-first international conference on Machine learning. p. 11. ACM (2004)
6. Christopher, D.M., Prabhakar, R., Hinrich, S.: Introduction to information retrieval. An Introduction To Information Retrieval **151**(177), 5 (2008)
7. Davidson, I., Basu, S.: A survey of clustering with instance level. Constraints **1**, 2 (2007)
8. Dhillon, I.S., Guan, Y.: Information theoretic clustering of sparse cooccurrence data. In: Data Mining, 2003. ICDM 2003. Third IEEE International Conference on. pp. 517–520. IEEE (2003)
9. Fodeh, S., Punch, B., Tan, P.N.: On ontology-driven document clustering using core semantic features. Knowledge and information systems **28**(2), 395–421 (2011)
10. Jaidka, K., Mumick, S., Chhaya, N., Ungar, L.: The CL-Aff Happiness Shared Task: Results and Key Insights. In: Proceedings of the 2nd Workshop on Affective Content Analysis @ AAAI (AffCon2019). Honolulu, Hawaii (January 2019)
11. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
12. Manning, C.D., Raghavan, P., Schütze, H.: Scoring, term weighting and the vector space model. Introduction to information retrieval **100**, 2–4 (2008)
13. Mikolov, T., Grave, E., Bojanowski, P., Puhrsch, C., Joulin, A.: Advances in pre-training distributed word representations. In: Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018) (2018)
14. 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. pp. 3111–3119 (2013)
15. Munkres, J.: Algorithms for the assignment and transportation problems. Journal of the society for industrial and applied mathematics **5**(1), 32–38 (1957)
16. Nasrabadi, N.M.: Pattern recognition and machine learning. Journal of electronic imaging **16**(4), 049901 (2007)
17. Wang, Z., Mi, H., Ittycheriah, A.: Semi-supervised clustering for short text via deep representation learning. arXiv preprint arXiv:1602.06797 (2016)
18. Xu, J., Wang, P., Tian, G., Xu, B., Zhao, J., Wang, F., Hao, H.: Short text clustering via convolutional neural networks. In: VS@ HLT-NAACL. pp. 62–69 (2015)