Exploring Kernel Functions in the Softmax Layer for Contextual Word Classification

Yingbo Gao, Christian Herold, Weiyue Wang, Hermann Ney

Human Language Technology and Pattern Recognition Group
Computer Science Department
RWTH Aachen University
D-52056 Aachen, Germany
<surname>@i6.informatik.rwth-aachen.de

Abstract

Prominently used in support vector machines and logistic regressions, kernel functions (kernels) can implicitly map data points into high dimensional spaces and make it easier to learn complex decision boundaries. In this work, by replacing the inner product function in the softmax layer, we explore the use of kernels for contextual word classification. In order to compare the individual kernels, experiments are conducted on standard language modeling and machine translation tasks. We observe a wide range of performances across different kernel settings. Extending the results, we look at the gradient properties, investigate various mixture strategies and examine the disambiguation abilities.

1. Introduction

With neural networks, tasks such as language modeling (LM) and machine translation (MT) are generally approached by factorizing the target sentence probability into products of target word posterior probabilities [1, 2, 3]. In order to classify over the target vocabulary, it is necessary to compute a context vector, learn a projection matrix and normalize the similarity scores between the two probabilities. While various model architectures are proposed to calculate the context vectors [4, 5, 6], most of them use a softmax layer with the inner product function to compute the word posterior probabilities. [7] identify a shortcoming with the formulation above which they call the "Softmax Bottleneck". The problem lies in the exponential-and-logarithm calculation when using the cross-entropy criterion, which results in a low-rank log word posterior probability matrix. Hypothesizing that natural language is high-rank, the authors argue that the "Softmax Bottleneck" is a limiting factor of the expressiveness of the models. One natural thought on this problem is its similarity to the lack of expressiveness of a logistic regression model or a support vector machine (SVM) with a simple linear kernel.

By implicitly transforming data points into high dimensional feature spaces, kernels can increase the expressiveness of the classifier and allow for more complex decision boundaries [8]. Note that a kernel is deemed valid when it corresponds to a scalar product in some feature space [8], or its corresponding Gram matrix is positive semidefinite [9]. Yet empirical results [10, 11] also show that conditionally positive semidefinite kernels can perform well in some applications. In this work, we do not enforce the positive semidefiniteness of kernels.

Motivated to examine the performances of various kernels in LM and MT, we structure this work as follows:

1. We implement individual kernels in replacement of the inner product function in the softmax layer and test them on LM and MT tasks.
2. We look at the gradient properties of several kernels and analyze the observed performance differences.
3. We investigate various mixtures of kernels.
4. We further examine and compare the disambiguation abilities of the linear kernel and a mixture of kernels.

2. Related Work

The softmax layer with the inner product similarity function has limits in terms of expressiveness: [7] identify the "Softmax Bottleneck", demonstrating its incapabilities to represent arbitrary target distributions. As a solution, they propose the "Mixture-of-Softmaxes" (MoS) architecture. [12] reanalyze the problem and suggest to include an extra sigmoid function in the softmax formula. [13] develop weight norm initialization and normalization methods on top of MoS. [14] extend the architecture and introduce a regularization term to encourage equal contributions of mixture components.

Kernels are generally considered to be a family of energy functions, which can implicitly map data points into high dimensional spaces, allowing for the learning of complex decision boundaries [8]. [15] provide detailed and extensive information on the topic of learning with kernels using SVMs. [16] curates an incomplete list of popular kernels. [17] build on kernel logistic regression and develop a classification algorithm called import vector machine. [18] explore the use of arccosine kernels in a multilayer nonlinear transformation setup. [19] introduce a vector of binary latent variables and propose to use a bilinear scoring function in the softmax.

In pursuit of more powerful word representations, [20] and [21] propose to embed words into Gaussian distributions to better capture entailment properties and multiple meanings of words. [22] show that an n-dimensional Poincaré ball is a suitable space, in which one can embed words to better represent hierarchies. [23] describe a re-parametrization trick to automate the process of renormalizing word vector norms.

3. Methodology

3.1. Generalized Softmax

According to [12], because of the "logarithm of exponential" calculation in the "softmax and cross entropy" setup, the non-linearity of the logarithm of the activated logit is a prerequisite to break the "Softmax Bottleneck". While the paper presents a Sigsoftmax activation function applied on logits calculated with inner products, we explore many non-linear kernel functions for the logit calculation, including the ones traditionally used in SVMs.

Specifically, we use a generalized softmax layer

$$\mathrm{p}(w_v|h) = \sum_{k=1}^{K} \pi_k \frac{\exp(\mathrm{S}_k(W_v, \tilde{h}_k))}{\sum_{v'=1}^{V} \exp(\mathrm{S}_k(W_{v'}, \tilde{h}_k))}, \quad (1)$$

with W_v being the v-th column of the projection matrix W and $\tilde{h}_k$ being the k-th transformed context vector. W is shared across K mixture components and each component uses kernel S_k to calculate the logits. Both the mixture weight π_k

$$\pi_k = \frac{\exp(M_k^{\mathrm{T}} h)}{\sum_{k'=1}^{K} \exp(M_{k'}^{\mathrm{T}} h)} \quad (2)$$

and the transformed context vector $\tilde{h}_k$

$$\tilde{h}_k = \tanh(C_k^T h) \quad (3)$$

depend on the original context vector h. In this setup, matrix $W \in \mathbb{R}^{d \times V}$, $M \in \mathbb{R}^{d \times K}$ and $C_k \in \mathbb{R}^{d \times d}$ are all trainable model parameters, where d is the hidden dimension size.

There are two main motivations behind this generalized setup: first, by mapping h to $\tilde{h}_k$, we hope to transform the context vector into the respective feature space and generate different logit distributions over the vocabulary; second, by explicitly conditioning π_k on h, we hope the model is able to select which kernel is more appropriate for each context. Note that, in Equation 1, W_v does not have a subscript of k, which means we tie the projection matrices across the kernels. This greatly limits the expressiveness of our model, but is a compromise because of memory limitations.

3.2. Individual Kernels

In total, we implement and experiment with 9 individual kernels $\mathrm{S}(W_v, h)$ – linear (`lin`), logarithm (`log`), power (`pow`), polynomial (`pol`), radial basis function (`rbf`), symmetric spherical Gaussian (`ssg`) [20], symmetric spherical mixtures of Gaussian (`mog`) [21], non-parametric hyperbolic (`hpb`) [23] and wavelet (`wav`) [24]:

$$\mathrm{S}_{\texttt{lin}} = W_v^T h, \quad (4)$$

$$\mathrm{S}_{\texttt{log}} = -\log(||W_v - h||^p + 1), \quad (5)$$

$$\mathrm{S}_{\texttt{pow}} = -||W_v - h||^p, \quad (6)$$

$$\mathrm{S}_{\texttt{pol}} = (\alpha W_v^T h + c)^p, \quad (7)$$

$$\mathrm{S}_{\texttt{rbf}} = \exp(-\gamma||W_v - h||^2), \quad (8)$$

$$\mathrm{S}_{\texttt{ssg}} = \log \int N(\mu_{W_v}, \Sigma_{W_v}) N(\mu_h, \Sigma_h), \quad (9)$$

$$\mathrm{S}_{\texttt{mog}} = \sum_{i,j} \log \int N(\mu_{i,W_v}, \Sigma_{i,W_v}) N(\mu_{j,h}, \Sigma_{j,h}), \quad (10)$$

$$\mathrm{S}_{\texttt{hpb}} = -\mathrm{acos}(1 + \frac{2||W_v - h||^2}{(1 - ||W_v||^2)(1 - ||h||^2)}), \quad (11)$$

$$\mathrm{S}_{\texttt{wav}} = \cos(\frac{||W_v - h||^2}{a}) \exp(\frac{-||W_v - h||^2}{b}). \quad (12)$$

These individual kernels can all be thought of as energy functions between the context vector h and the word vector W_v. Because of the exponential calculation outside of the logit calculation, these kernels may result in numerically unstable computations. For example, using the `rbf` kernel results in an exponential-of-exponential operation, which easily blows up when W_v and h are distant. We nonetheless implement and examine the properties of these kernels.

Additionally, the memory consumption may blow up when using certain kernels. This is because the dimension reduction step in d common to all kernels may not always be immediately executable. In this case, all pairwise similarities/distances between the context vectors and the word vectors have to be cached. To reduce memory usage, we apply several tricks: 1. use spherical covariance matrices, 2. simplify the wavelet kernel and 3. rewrite the formula of the power of the vector difference norm

$$||W_v - h||^p = (||W_v||^2 + ||h||^2 - 2W_v^{\mathrm{T}} h)^{\frac{p}{2}}, \quad (13)$$

which also suggests that $||W_v - h||^p$ can be thought of as a vector norm regularized version of the inner product.

4. Experiments

4.1. Experimental Setup

In this work, two datasets are used: Switchboard (SWB) for LM and IWSLT 2014 German→English (IWSLT) for MT. SWB is a relatively small dataset, with a vocabulary size of 30k and a training token count of 25M. For SWB, we use a standard 2-layer LSTM to generate context vectors, with 512 hidden dimensions and 0.1 dropout on the embedded word vectors. For IWSLT, we follow the setup in [25], using 160k parallel training sentences and 10k joint BPE merge operations. The transformer architecture is used to produce context vectors. We use 512 hidden dimensions in the encoder and decoder stacks, 1024 hidden dimensions in the fully-connected layers and 4 attention heads. As in Equation 1,

the context vectors are compared with the word vectors in the projection matrices. Hyperparameters of the kernels are tuned with grid search to give the best performance on the development set. We vary K and S_k to test various kernel settings. We use the Fairseq toolkit [26] to conduct the experiments.

4.2. Individual Kernels

The performances of models using individual kernels are summarized in Table 1. References from the literature are included to show the relative strengths of the kernels.

Method	SWB (PPL)	IWSLT (BLEU[%])
Ref.	[27] 47.6	[28] 35.2
lin	46.8	34.3
log	103.0	0.4
pow	46.8	32.8
pol	47.3	31.7
rbf	284.9	0.0
ssg	49.9	34.6
mog	46.7	34.2
hpb	122.6	0.3
wav	289.7	0.0

Table 1: Performance of individual kernels.

Compared to the lin kernel, all other individual kernels have the exact same number of parameters and comparable run time. The only difference lies in how the logits are calculated. On both datasets, we see consistent behavior. While lin serves as a reasonably good baseline, pow, pol, ssg and mog are on the same level of performance, even slightly outperforming lin in some cases (mog on SWB and ssg on IWSLT). log and hpb are worse, giving much higher perplexity (PPL) and values close to zero in BLEU[%] [29]. Among all 9 kernels, rbf and wav perform the worst.

4.3. Gradient Properties

As shown in Section 4.2, a wide range of performances is observed across different kernels. In order to understand why some kernels perform better than others, we select four simple kernels (rbf, wav, log and pow) and plot their function graphs in Figure 1.

All kernels have their maximum values at $||W_v - h||^p = 0$. In this case, the context vector h is exactly the same as the word vector W_v. The gradient properties, however, vary across these kernels. When far away from the optimum, pow has a constant non-zero gradient. On the other hand, rbf, wav and log have near-zero gradients. As $||W_v - h||^p$ approaches zero, the absolute gradient of log increases, while non-negligible gradients show up in rbf and wav only when $||W_v - h||^p$ is close to zero. We think strong supervised signals in the gradients are helpful for model convergence. Considering these gradient properties, we expect performances among these kernels to be: rbf $\approx$ wav $<$ log $<$ pow. The results in Table 1 fits our expectations very well. This further suggests that when selecting and designing alternative kernels, the gradient properties across the domain of the parameters should be carefully considered.

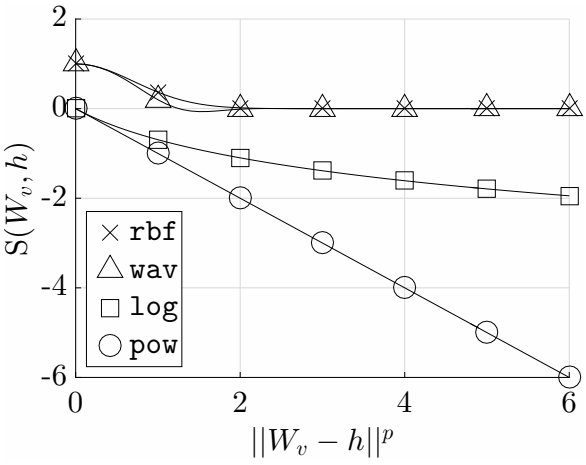

Figure 1: Graphs of rbf, wav, log and pow ($p = 2$).

4.4. Mixtures of Kernels

Inspired by the MoS approach, we train LMs combining the outputs of multiple kernels according to Equation 1 on SWB. Similar to [14], we add the variance of the mixture weights, scaled by ρ and averaged over data N, to the standard cross entropy loss:

$$L_{\text{reg}} = L_{\text{ce}} + \frac{\rho}{N} \sum_{N} \text{Var}_N(\pi_k) \tag{14}$$

ρ	Variance	PPL
0.001	4.74	46.8
0.01	4.98	46.6
0.1	3.67	47.2
1	3.81	47.4

Table 2: Regularization of π.

Name	Mixture settings	PPL
mos	9×lin	47.8
mix_{big}	1 of each kernel	47.1
mix_1	lin, log, rbf, hpb, wav	46.6
mix_2	3×lin, log	46.5
mix_3	lin, log, pow, pol	47.3
mix_4	lin, log, rbf, hpb	47.1
mix_5	2×lin, 2×rbf	46.7

Table 3: Performance of mixtures of kernels.

Model	Prediction
Ground Truth	... books can end up being outdated very **quickly**
`lin`	... books can end up being outdated very **soon**
mix_{big}	... books can end up being outdated very **quickly**
Ground Truth	... if you vote for a republican or vote for a **democrat**
`lin`	... if you vote for a republican or vote for a **republican**
mix_{big}	... if you vote for a republican or vote for a **democrat**

Table 4: Some examples of the disambiguation abilities of `lin` vs mix_{big}.

Performances of MoS systems for different values of ρ are depicted in Table 2. We decide to run all mixture experiments with $\rho = 0.1$, as it seems to be a good compromise between regularization and performance.

The detailed mixture settings and perplexity results are summarized in Table 3. Specifically, we select "mos" to try to reproduce the "Softmax Bottleneck" paper [7] and "mix_{big}" to test a big mixture of each kernel. "mix_1", "mix_2", ..., and "mix_5" are selected randomly to explore the kernel combination space. We also experiment with more mixture settings, but unfortunately with tied projection matrices, only those mixtures with the `lin` kernel give good performance. Note that weighted matrices are tied and multiple instances of the same kernel may be included in a mixture component. In this case, each mixture component is free in learning its own context vectors.

Compared to the individual kernels, the decoding speed of the mixture models is slowed down by a factor of two on average. The increased number of parameters because of context vector projection is negligible when the projection matrices are tied. As can be seen, all the mixture settings in Table 3 have similar performances to the simple `lin` setup in Table 1. This is very likely because they all have at least one linear component, and the linear components consistently receive a total weight above 50%. So we conclude that mixtures of kernels using a shared projection matrix cannot significantly improve over the baseline. We find no fundamental difference between the open-sourced "Mixture-of-Softmaxes" implementation [7] and ours. Unfortunately, we can not replicate the results from the original paper. We do note that they use different datasets and include many more techniques like activation regularization and averaged SGD optimization.

4.5. Disambiguation Abilities

In theory, there is a potential drawback of the `lin` kernel used together with the softmax layer. Consider when two words v_1 and v_2 are close syntactically and/or semantically. It is a common observation that their corresponding word vectors are also close together after successful training [30, 31, 32]. In this case, for any context vector h, the logits $W_{v_1}^T h$ and $W_{v_2}^T h$ will be similar as well. Although the alternative kernels studied here also suffer from this problem: $S(W_{v_1}, h) \approx S(W_{v_2}, h)$ when $W_{v_1} \approx W_{v_2}$, with non-linear activations the difference between the logits may be amplified, making it easier to disambiguate the words.

To show potentially better disambiguation properties of kernel mixtures, we take a more detailed look at the LM task. For the `lin` model, the projection matrix is extracted and the pairwise word distances are calculated using inner product. This is then used to extract word clusters in the embedding space. Two of the extracted clusters are: {`quickly`, `slowly`, `soon`, `quick`, `easily`} and {`republicans`, `politicians`, `democrat`, `republican`, `democrats`}. We suspect that it might be difficult for the `lin` model to distinguish words in these clusters, as their similarity scores are very close. It turns out that this is also what we observe when looking at the example sentences shown in Table 4. This suggests that even if diversifying the output layer with different kernels does not result in immediate improvements in terms of perplexity – a kernel-mixture-based method may still be superior in other aspects.

5. Conclusion

Motivated by the similarity between the "Softmax Bottleneck" problem and the lack of expressiveness of a logistic regression model or an SVM with a simple linear kernel, we explore the use of kernel functions in the softmax layer for contextual word classification:

1. In replacement of the inner product function, kernels and mixtures of kernels are used in the softmax layer. Our experiments with 9 different individual kernels on LM and MT exhibit a wide range of performances, with `lin`, `pol`, `pow`, `ssg` and `mog` being the best-performing ones.

2. Examining the gradient properties, we give reasons why some kernels perform better than others and argue that the gradient properties of a kernel function across the domain of the parameters is worthy of careful consideration.

3. In mixture settings consisting of at least one `lin` kernel, `lin` consistently receives a large weight.

4. While not significantly better than the `lin` kernel, we observe cases where the mixture model is better at disambiguating similar words.

In our mixture experiments, projection matrices are shared due to memory constraints. This greatly limits the expressiveness of the model. The next step is to untie the word embeddings across different kernels and allow for the learning of even more complex decision boundaries.

6. Acknowledgements

This work has received funding from the European Research Council (ERC) (under the European Union's Horizon 2020 research and innovation programme, grant agreement No 694537, project "SEQCLAS") and the Deutsche Forschungsgemeinschaft (DFG; grant agreement NE 572/8-1, project "CoreTec"). The GPU computing cluster was supported by DFG (Deutsche Forschungsgemeinschaft) under grant INST 222/1168-1 FUGG.

7. References

[1] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *Journal of machine learning research*, vol. 3, no. Feb, pp. 1137–1155, 2003.

[2] M. Sundermeyer, R. Schlüter, and H. Ney, "Lstm neural networks for language modeling," in *Thirteenth annual conference of the international speech communication association*, Portland, OR, USA, 2012, pp. 194–197.

[3] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[4] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *3rd International Conference on Learning Representations (ICLR)*, San Diego, CA, USA, 2015.

[5] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, "Convolutional sequence to sequence learning," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 1243–1252.

[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008.

[7] Z. Yang, Z. Dai, R. Salakhutdinov, and W. W. Cohen, "Breaking the softmax bottleneck: A high-rank rnn language model," in *Sixth International Conference on Learning Representations (ICLR)*, Vancouver, Canada, 2018.

[8] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[9] J. Shawe-Taylor, N. Cristianini, *et al.*, *Kernel methods for pattern analysis*. Cambridge university press, 2004.

[10] H.-T. Lin and C.-J. Lin, "A study on sigmoid kernels for svm and the training of non-psd kernels by smo-type methods," Tech. Rep., 2003.

[11] S. Boughorbel, J.-P. Tarel, and N. Boujemaa, "Conditionally positive definite kernels for svm based image recognition," in *2005 IEEE International Conference on Multimedia and Expo*. IEEE, 2005, pp. 113–116.

[12] S. Kanai, Y. Fujiwara, Y. Yamanaka, and S. Adachi, "Sigsoftmax: Reanalysis of the softmax bottleneck," in *32nd Conference on Neural Information Processing Systems (NeurIPS)*, Montréal, Canada, 2018.

[13] C. Herold, Y. Gao, and H. Ney, "Improving neural language models with weight norm initialization and regularization," in *Proceedings of the Third Conference on Machine Translation: Research Papers*, 2018, pp. 93–100.

[14] S. Takase, J. Suzuki, and M. Nagata, "Direct output connection for a high-rank language model," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018, pp. 4599–4609.

[15] B. Schölkopf, A. J. Smola, F. Bach, *et al.*, *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[16] C. R. Souza, "Kernel functions for machine learning applications," Available at http://crsouza.com/2010/03/17/kernel-functions-for-machine-learning-applications/(2019/10/10).

[17] J. Zhu and T. Hastie, "Kernel logistic regression and the import vector machine," in *Advances in neural information processing systems*, 2002, pp. 1081–1088.

[18] Y. Cho and L. K. Saul, "Kernel methods for deep learning," in *Advances in neural information processing systems*, 2009, pp. 342–350.

[19] R. Memisevic, C. Zach, M. Pollefeys, and G. E. Hinton, "Gated softmax classification," in *Advances in neural information processing systems*, 2010, pp. 1603–1611.

[20] L. Vilnis and A. McCallum, "Word representations via gaussian embedding," in *3rd International Conference on Learning Representations (ICLR)*, San Diego, CA, USA, 2015.

[21] B. Athiwaratkun and A. G. Wilson, "Multimodal word distributions," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vancouver, Canada, 2017, pp. 1645–1656.

[22] M. Nickel and D. Kiela, "Poincaré embeddings for learning hierarchical representations," in *Advances in neural information processing systems*, 2017, pp. 6338–6347.

[23] B. Dhingra, C. Shallue, M. Norouzi, A. Dai, and G. Dahl, "Embedding text in hyperbolic spaces," in *Proceedings of the Twelfth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-12)*, 2018, pp. 59–69.

[24] L. Zhang, W. Zhou, and L. Jiao, "Wavelet support vector machine," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 34, no. 1, pp. 34–39, 2004.

[25] S. Edunov, M. Ott, M. Auli, D. Grangier, and M. Ranzato, "Classical structured prediction losses for sequence to sequence learning," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, New Orleans, LA, USA, 2018, pp. 355–364.

[26] M. Ott, S. Edunov, A. Baevski, A. Fan, S. Gross, N. Ng, D. Grangier, and M. Auli, "fairseq: A fast, extensible toolkit for sequence modeling," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, Minneapolis, MN, USA, 2019, pp. 48–53.

[27] K. Irie, Z. Lei, L. Deng, R. Schlüter, and H. Ney, "Investigation on estimation of sentence probability by combining forward, backward and bi-directional lstm-rnns," *Proc. Interspeech 2018*, pp. 392–395, 2018.

[28] F. Wu, A. Fan, A. Baevski, Y. N. Dauphin, and M. Auli, "Pay less attention with lightweight and dynamic convolutions," in *Seventh International Conference on Learning Representations (ICLR)*, New Orleans, LA, USA, 2019.

[29] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, 2002, pp. 311–318.

[30] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[31] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *First International Conference on Learning Representations (ICLR)*, Scottsdale, AZ, USA, 2013.

[32] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *International conference on machine learning*, 2014, pp. 1188–1196.