Exploring the Adaptability of Word Embeddings to Log Message Classification

Yusufu Shehu
Moogsoft Ltd
River Reach, 31-35 High Street
Kingston-Upon-Thames, UK
yusufu.shehu@moogsoft.com

Robert Harper
Moogsoft Ltd
River Reach, 31-35 High Street
Kingston-Upon-Thames, UK
rob@moogsoft.com

***Abstract*—Minimizing the resolution time of service-impacting incidents is a fundamental objective of IT operations. Enriching the meta-data of the events and logs ingested by such systems using AI-based classifiers greatly increases the efficacy of features such as root cause analysis and workflow automation, and hence reduces incident remediation time.**

The use of word embeddings in text classification tasks is well-established, however, the general English corpora used to generate off-the-shelf embeddings lack the domain-specific lexicon required for accurate classification of event and log data. In the current contribution, we investigate multiple ways in which this deficiency can be addressed. In addition to augmenting the training-corpus with a domain-specific lexicon, we increase the granularity of our embedding using character *n*-gram decompositions and sub-word level representations. All implementations improved classification accuracy over the base case. Further, we explore the performance of a sequence classifier with embeddings of varying domain specificity. We observe that the performance of high-specificity models reduces as the volume of previously unseen words in the test data increases. We conclude that for a multi-input use case, and by leveraging sub-word level information, a high-specificity model can be outperformed by a model trained on a low-specificity corpus.

I. Introduction

The primary objective of IT operations is to minimize the time required to resolve service-impacting incidents. Fault localization, [1], [2] is the process of deducing the source of a failure from a set of indicators and is fundamental to achieving this objective. In turn, an important enabler of efficient fault localization is data enrichment, a process whereby inbound events and log messages are augmented with meta-data that can further facilitate the remediation process.

Traditionally, configuration management databases have been the primary source of enrichment data, however, these systems are costly to create, costly to maintain, and can quickly become outdated. AI-based text-classification techniques offer a new approach to data enrichment. By analyzing inbound events and log messages, pre-defined labels from multiple categories can be automatically assigned to provide additional indicators for root cause analysis, or task assignment for example.

State-of-the-art text-classification techniques rely heavily on concepts such as sequence modelling and word-embeddings. Word embeddings, [3], [4], [5] are typically used as an input to text-classification tasks because they provide semantically rich representations in low-dimensional feature spaces. They are operational applications of the distributional hypothesis in linguistics, which states that semantically similar words occur in similar contexts. An embedding for a word is a vector of real-numbers derived from the internal weights of a neural network that implicitly learns word co-occurrences in a body of training text. Natural language processing (NLP) tasks have benefited hugely from these techniques, thanks largely to pre-trained embeddings learned from general-language corpora comprising billions of words. In applications that rely upon a domain-specific lexicon, general-language word embeddings suffer from a large number of missing representations. Unknown word vectors offer no useful information and consequently the performance of pre-trained embeddings is reduced in these cases. The application of NLP techniques to managing compute and communication infrastructures is one such scenario.

In previous work [6], we described the process of incrementally enriching word embeddings with domain-specific words and described the topology of clusters of domain-specific word vectors as a function of increasing corpus enrichment. We explore the success of word embedding enrichment for log message classification and explore multiple approaches to improve word-level representations for machine learning.

We have two aims in this paper, firstly to explore if there is a performance benefit for log message classification by enhancing word-level features. The enhancement can be produced by using sub-word level representations through either the feature hashing of character *n*-grams, increasing the number of domain-specific word vectors in the word embedding, or combining the word embedding with character-level encodings. Secondly, we aim to explore the impact of varying the training corpus domain-specificity on the log message classification performance accuracy. We use three ablation experiments to determine how the varying levels of domain-specific word-level information impact the robustness of the internal weights learned from the word and character-level inputs.

This paper begins in Section II by outlining the previous work and recent developments in word embeddings and text-

978-3-903176-32-4 © 2021 IFIP

based feature extraction for sequence classification. Section III describes the multi-modal sequence classifier model used in the analysis. We present the first analysis deriving alternatives to word embeddings together with the results in Section IV. The second analysis, investigating word embedding coverage and classifier performance is presented in Section V. We make our concluding points and present the next steps for further research in Section VI.

II. Previous Work

To overcome the problem of missing domain-specific words in embeddings, steps have been made to fine-tune pre-trained embeddings. A word annotation embedding algorithm was proposed [7] to augment word embeddings of input texts from specialized domains. The method was evaluated on two cyber-security text corpora and was effective in learning domain-specific word embeddings. We explored [6] the fine-tuning of general English word embeddings with network infrastructure specific text. We modeled the evolution of specificity in word vectors by observing clusters of domain-specific words at incremental stages of word embedding enrichment. Using a combination of affinity propagation and principal component analysis, we observe domain-specific words of similar context group together, and their clusters become increasingly well separated as the training corpus is enriched incrementally with domain-specific sequences.

Recent innovations in embedding development have focused on bi-directionality and the attention mechanism to enhance learning context from text. Examples of these developments include ULMFiT [8], BERT [9], and ELMo [10], which are used for general domain text. BioBERT [11], a domain-adapted BERT model outperformed the current embedding state-of-the-art BERT for domain-specific tasks by including domain-specific documents in the pre-training phase.

There have been recent developments in providing alternatives to common word embeddings. A hash embedding [12], represents a word by several *n*-dimensional embedding vectors and a weight vector, selected by the hashing trick. Hash embeddings can efficiently deal with corpora containing millions of words, and do not require a dictionary before training. These embeddings demonstrate at least the same performance level as regular word embeddings across a wide range of tasks.

Leveraging sub-word level information is a powerful solution to unknown words in embeddings. FastText [13] is an embedding method exploiting sub-word level information. Representations are learned for character *n*-grams, and words represented as the sum of the *n*-gram vectors. Encoding character level information reduces the dependency on word-level features and helps models understand suffixes and prefixes. This method achieves state-of-the-art performance on word similarity and analogy tasks. [14] proposed a novel neural network architecture that automatically combines word embeddings and character-level features to derive a concatenated representation of language data for named entity recognition.

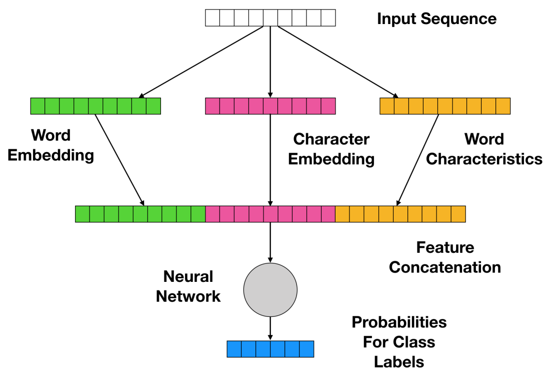


Fig. 1: Simplified diagram for multi-modal sequence classifier.

III. Model Description

Sequences of words have distinct levels of feature granularity that we can incorporate during model training. Log messages contain high proportions of words that do not commonly occur in natural language but have distinct arrangements of characters. For example, IPv4 addresses have a consistent, internal structure of numerical and special characters (dots). If we rely solely on word-level features, the model is unable to exploit these character-level morphologies. We encode word syntax and semantics using a word embedding, character-level syntax using a character embedding, and one-hot vectors to encode character contents in a word. We use an adaptation of the hybrid bidirectional LSTM and CNN architecture, which has been previously used for named entity recognition [14]. This model takes the three representations as input, with a convolution operation performed on the character embedding before concatenation. After feature concatenation, the vector is fed into a bidirectional LSTM and a dense layer to produce probabilities across the training labels. We show a simplified diagram of the information flow through the model from input sequence to class label probabilities in Figure 1.

IV. Analysis I: Alternatives to Word Embeddings

The model defined in Section III is trained on log messages, which contain words not typically found in generic English word embeddings. Furthermore, the log messages are generally sparse, with domain and environment-specific words occurring only a small number of times. An example of a log message is shown in Figure 2, we can see the important words for classification are mostly domain specific: *router2*, *UI_DBASE_LOGIN_EVENT*. The other words are identifiers and time-specific words (date, time) whose variations do not introduce any further information to learn from, e.g., “joe”. A general word embedding will have low representation for already feature sparse sequences.

```
user@host> show log messages May 20 19:30:25 router2 mgd[8097]:
UI_DBASE_LOGIN_EVENT: User 'joe' entering configuration mode.
```

Fig. 2: Example of a log message.

TABLE I: Model performance, word embedding included.

Label	Prec.	Rec.	F1	Accuracy (%)
Application	0.97	0.85	0.90	0.98
Network	1.00	0.99	0.99	
OS	0.97	0.94	0.96	
Server	0.99	1.00	0.99	
Storage	0.97	0.93	0.95	
Unknown	0.87	0.98	0.92	

TABLE II: Model performance, word embedding not included.

Label	Prec.	Rec.	F1	Accuracy (%)
Application	0.83	0.68	0.75	0.89
Network	0.99	0.96	0.98	
OS	0.78	0.73	0.75	
Server	0.97	0.87	0.92	
Storage	0.60	0.49	0.54	
Unknown	0.53	0.90	0.67	

In this analysis, we ablate the word embedding from the model and explore embedding alternatives that minimize or remove the occurrences of unknown words. We postulate that the observations from this analysis will give us insight into how useful general word embeddings can be, the impact of enriching them with domain-specific words, and the availability and usefulness of word embedding alternatives.

A. *Performance with Word Embedding Removed*

The ablation study involves removing the word embedding from the classifier while keeping the character embedding and word characteristics. The labels used categorize six types of activity commonly logged by IT infrastructure. We compared the original model's log message classification performance with the ablated model, as shown in Table I and Table II respectively. The ablation of the word embedding results in a drop in accuracy from 0.98 (Table I) to 0.89 (Table II). This 9% drop in performance is significant and indicates the importance of having a word-level feature for the classifier to learn optimally.

B. *Performance with Alternative Word-level Representations*

Exploring alternative word-level representations can minimize or avoid unknown words during training, which can improve model performance. We define and test two approaches, a hash embedding method and a composite n-gram vector method. The hash embedding prescription is described as follows. For every word, decompose the word into a set of character n-grams. We define a vector for each character n-gram get $hash(n - \text{gram})\ mod\ k$ where k is the size of the collection of character n-grams. The result is a look-up index with the key being the character n-gram and value is the hash value.

The composite n-gram vector method involves summing the one-hot vectors with values indexed by character n-gram hash values. For every word in a tokenized log message, we obtain a set of character n-grams. We obtain the hash as a sequence of bytes for each character n-gram and calculate the modulo with hash size to get the hash index. Each character n-gram has a one-hot vector, and the hash index specifies which element to set to 1. Consequently, each word has a vector, which is the sum of the character n-gram one-hot vectors. For this analysis, we set n to equal 3.

TABLE III: Model performance for different word features.

Model	Accuracy (%)
Word Embedding	97.7
Hash Embedding	**98.3**
Shingle Hash Trick	96.0

1) Results: Referring to Table III, we find that using the hash embedding as word-level input to the classifier produces better performance than the original word embedding.

The hash embedding approach introduces a shared pool of embedding vectors for n-grams selected by the hashing trick. By providing n-gram vectors, the feature granularity during training increases, and the model has exposure to more representations of words in the training corpora. The vectors extracted also belong to denser feature space (lower dimensionality) compared to regular word embeddings. Sparser features are generally more challenging to learn. However, with the hash embedding, there can still be instances of unknown n-grams.

The composite n-gram method can eliminate the occurrence of unknown words or character n-grams. During the analysis, we observed that the classification performance increased as the length of the composite n-gram vectors increased. While hash-based techniques minimize or avoid unknown words, the resulting features are based more on lexical similarity than semantic. In contrast, common word embeddings are based on semantic and syntactic similarity. Encoding of semantic and syntactic information generally produces richer feature sets to learn from, enabling models to be more adaptable and maintain high performance for long sequences [10].

V. Analysis II: Varying Word Embedding Coverage

A. *Enriching Word Embeddings with Domain-Specific Words*

In this analysis, we define the metric coverage to quantify unknown words concerning the word embedding. Coverage, (Cov_{global}), is defined as:

$$Cov_{\text{global}} = 100 * \frac{w_k}{(w_k + w_u)}, \tag{1}$$

where w_k is the number of known words and w_u is the number of unknown words. Note that the word counts used are not the number of unique words but a count across all training data. We use global coverage to understand how many domain-specific words, *in all contexts*, that the model is exposed to during training. Multiple instances of domain-specific words will improve the learning of semantics by the model; by counting unique words, we ignore the importance of multiple contexts. However, we recognize counting domain-specific words as they occur once and only once can give a simplified representation of the specificity of a word embedding. We define Cov_{log} as the coverage of a given log message during testing. Cov_{log} is dependent on Cov_{global} as the embedding used during training is used to count how many known and

TABLE IV: Model performance with varying embedding coverage.

Corpus	Coverage (%)	Accuracy (%)
GloVe	54.7	97.7
WT-103	41.4	96.8
WT-103 + Stack-Ex	54.5	97.3
WT-103 + Stack-Ex + Cis	54.6	98.2
WT-103 + Stack-Ex + Cis + Service Environment Logs	89.6	98.7

unknown words there are in an individual log message during testing. These definitions reflect the possibility of different coverage values at training and testing.

1) Dataset Description: We use the same datasets considered in [6]: Wikitext-103 (*WT-103*) [15], Stack Exchange (*Stack-Ex*) [16], and Cisco (*Cis*) [17] to demonstrate an increasing gradient of lexicon specificity. We consider *WT-103* as a general English embedding, *Stack-Ex* as a general information technology embedding, and *Cis* as a highly specific information technology embedding. Each dataset is a truncation of an original corpus to 10,000,000 words to represent most words available while minimizing computing resources during experimentation. We also include the GloVe 6B-word embedding as a second example of a general English embedding that is widely used for NLP tasks. To maximize specificity, we created a corpus consisting of log messages extracted from multiple computing service environments. We refer to this corpus as *Service Environment Logs*. The corpus was built using log messages from six distinct environments giving approximately 1,200,000 additional domain-specific words. We combine the considered corpora to increase the number of domain-specific words available in the word embedding.

2) Results: We refer to Table IV to compare the coverage and accuracy of the models trained with the two general English corpora, GloVe and WT-103. We observe that GloVe has the highest coverage with 54.7%. Despite the difference in coverage of 13.3% between GloVe and *WT-103*, the difference in accuracy is 0.9%. We observe an incremental improvement in classification accuracy with increasing coverage, with an overall accuracy increase of 0.5% between the three corpora model (*WT-103* + *Stack-Ex* + *Cis*), and the model trained with the three corpora plus *Service Environment Logs*. We state that the best-case scenario for sequence classification is to pre-train a word embedding with a corpus containing relevant service environment language to maximize coverage during training. However, we observe diminishing returns in classification accuracy by increasing our coverage from approximately 50% to 90%. Referring back to Section III, the model takes sub-word level information to learn how to classify log messages correctly. The performance of the model is shared across all inputs rather than the word embedding exclusively. As a result, the performance increase with increasing coverage is attenuated. For a word-embedding only model, we may observe more linear increases in performance.

B. Varying Coverage with Feature Ablation

The classifier model takes three inputs to classify log messages: word embedding, character embedding and word characteristics one-hot vectors. We assess the impact of input feature ablation on classifier performance under different training coverage, Cov_{global}, conditions. We have two corpus configurations that give us two variations of coverage:

- High coverage ($Cov^{\text{H}}_{\text{global}}$) - Consisting of *WT-103*, *Stack-Ex*, *Cis*, and *Service Environment Logs* give 89% coverage.
- Low coverage ($Cov^{\text{L}}_{\text{global}}$) - Consisting of *WT103* truncated to 10% of its original size with the word frequency in the corpus increased from >0 to >1000.

The three ablations of the classifier model is:

- No word embedding (*no_word_emb*).
- No character embedding (*no_char_emb*).
- No word characteristics (*no_word_char*).

We aim to assess which ablation of the model is most affected by the training coverage. We varied the learning rate to test the hypothesis that different inputs to the model are learned at different rates either due to the complexity or length of the feature.

1) Results: We show the overall accuracy for the different model ablations in Fig. 3. Overall, the best model performance is exhibited with all three inputs. Removing a high coverage word embedding leads to the most significant drop in performance, and the variation of learning rate has minimal impact on the drop. If the word embedding coverage is low, removing the character embedding produces the most significant reduction in performance, particularly when the learning rate is 1×10^{-3}. The low coverage during training has the effect of "turning on" the weights for the character-level features. The model benefits from learning the character embedding and word characteristics with a high learning rate than the word embedding. However, during the analysis, we observed that the gradient descent was less stable for the word embedding ablated model. The inclusion of the word embedding reduces the occurrence of vanishing or exploding gradients. This observation could be an artifact of the word vector's richness as a feature, which could improve the gradient descent path to local minima.

C. Impact of Varying Coverage on Classifier Performance

In operation, we can encounter log messages in a new service environment with a lexicon that has not been given representation by a previously trained embedding. We aim to quantify how well such a model performs on an unseen corpus of log messages and whether training a model with a low level of representation forces the model to rely on character-level representation during learning. This section explores the impact on performance for models with varying levels of Cov_{global} on the classification of test log messages with varying levels of Cov_{log}. It is important to reiterate that Cov_{log} is dependent on Cov_{global}, e.g., if the training coverage is low, then the number of words in the test log message for which the trained model has a representation will be low on average. We explored three training coverage environments Cov^{89}_{global}, Cov^{71}_{global}, and Cov^{9}_{global}, where the superscript numbers are the

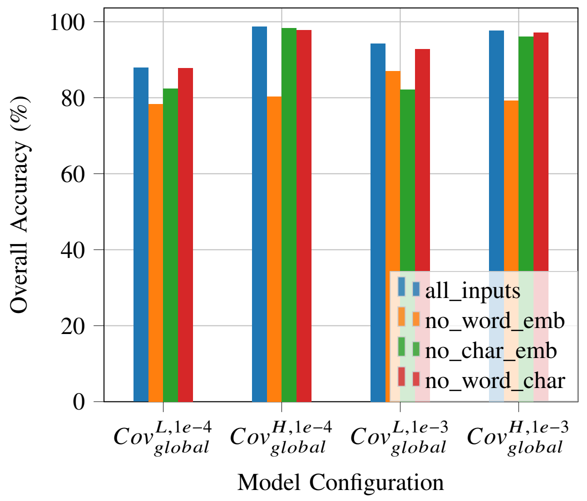
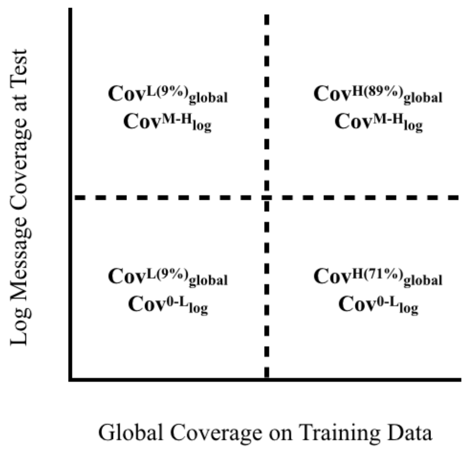


Fig. 3: Classification Performance with Feature Ablation under Cov^{H}_{global} and Cov^{L}_{global} conditions.



Log Message Coverage at Test
$Cov^{L(9\%)}_{global}$ $Cov^{M\text{-}H}_{log}$
$Cov^{H(89\%)}_{global}$ $Cov^{M\text{-}H}_{log}$
$Cov^{L(9\%)}_{global}$ $Cov^{0\text{-}L}_{log}$
$Cov^{H(71\%)}_{global}$ $Cov^{0\text{-}L}_{log}$
Global Coverage on Training Data

Fig. 4: Quadrant defining the four coverage conditions explored for classification performance analysis.

training coverages as percentages. These shorthand descriptors also refer to the model used for testing. Cov^{71}_{global} is an adapted high coverage model, used to represent the condition where an embedding has been enriched with domain-specific words but has low coverage for an unseen dataset. To create this, we excluded from the training dataset the single most populous service environment in terms of log message volume. However, we kept the log messages belonging to the excluded environment in the test dataset. By doing this, we ensure the model is still trained with an embedding that contains a broad range of domain-specific language but without removing any of the language unique to that environment.

This exclusion dropped the Cov_{global} from 89% to 71%, which we still consider to be high but increased the number for low coverage log messages during testing to a suitable level for analysis. We counted the number of correct and incorrect predictions for each description sequence and labeled each sequence by their Cov_{log} value. There are five ranges of coverage for the log messages during testing, which are considered in Table V. The *Zero* to *Low* bins are defined shorthand by as Cov^{0-L}_{log} and *Medium* to *High* are with Cov^{M-H}_{log}, as shown in Table V.

We define four coverage environments, which are each a combination of one training and one testing coverage environment. This is shown in Fig. 4. The 89% coverage model is used in the top-right quadrant of Fig. 4 as this model has both high training coverage and high coverage on the test log messages. The 71% coverage model is used in the bottom-right quadrant as this model has high training coverage and low coverage on the test log messages. The 9% coverage model is used in both the bottom-left and top-left quadrants as this model has low training coverage but has medium to high coverage on a sufficient number of the test log messages.

TABLE V: Ranges of test coverage from 0 to 100%.

Coverage Range (%)	Coverage Range Label	Shorthand
$Cov_{\text{log}} == 0$	Zero	Cov^{0-L}_{log}
$0 < Cov_{\text{log}} \leq 10$	Very Low	
$10 < Cov_{\text{log}} \leq 33.3$	Low	
$33.3 < Cov_{\text{log}} \leq 66.6$	Medium	Cov^{M-H}_{log}
$66.6 < Cov_{\text{log}} \leq 100$	High	

For this part of the analysis, we kept the learning rate fixed at 1×10^{-4}, as we have demonstrated the impact of learning rate variation on each feature in Section V-B. A learning rate of 1×10^{-4} favors the word embedding, which is the most important input to the model.

1) Results: Referring to Table VI, we observe that for the Cov^{9}_{global} model, the test coverage is significantly reduced, with no log messages classified in the *High* bin. Most of the correct predictions are made on low Cov_{log} sequences, including those with 0 coverage, which reinforces the need for the character embedding and word characteristics input during prediction. With both high global coverage models Cov^{89}_{global} and Cov^{71}_{global}, we observe significantly large statistics in the *Medium* to *High* bins. Based on the ablation experiments in Section V-B, we postulate that the character embedding and word characteristics weights are "tuned down", with most of the classification performance driven by the word embedding. Comparing the Cov^{71}_{global} with Cov^{89}_{global}, we see the Cov^{71}_{global} model produces significantly more statistics (the sum of both correct and incorrect classifications) to assess performance in the low test coverage region than the Cov^{89}_{global} model, e.g., two orders of magnitude greater in the *Low* bin. The Cov^{71}_{global} model correctly classifies the majority of low Cov_{log} sequences, particularly in the *Low* bin. The Cov^{71}_{global} model correctly classifies almost all of the alerts in the *Zero* bin. Comparing the Cov^{71}_{global} and Cov^{9}_{global} models, we observe the Cov^{71}_{global} model has fewer statistics in Cov^{0-L}_{log} region than the Cov^{9}_{global}, but has sufficient statistics to make valid comparisons. The Cov^{71}_{global} coverage model has similar performance to the Cov^{9}_{global} model but is less consistent in classifying sequences in the *Very Low* bin. We expect this, as the Cov^{9}_{global} model has its character embedding and word characteristic weights "tuned up", and

TABLE VI: Comparison of model performance with different pre-trained word embeddings.

Cov_{global} (%)	Cov_{log} (%)	**Incorrect Classification**	**Correct Classification**
9	Zero	7007	20983
	Very Low	8162	97857
	Low	12490	81924
	Medium	87	736
	High	0	0
71	Zero	14	208
	Very Low	166	331
	Low	185	3595
	Medium	2764	56512
	High	4405	161066
89	Zero	1	23
	Very Low	0	0
	Low	10	29
	Medium	201	4238
	High	2870	221874

so will be more consistent in performance when word-level information (coverage) is low in the test data. The Cov^{71}_{global} model is more consistent in the *Very Low* bin than the Cov^{9}_{global} model, getting only 5% of the sequences incorrectly labeled compared to 15%, although there are comparatively lower statistics. This result highlights that the Cov^{71}_{global} model can make use of low to medium coverage and consistently classify.

Overall, the results indicate that using a low training coverage forces the model to use character-level information to correctly classify log messages. A low coverage model can be more adaptable to unseen service environments. It does not rely heavily on the domain-specific word-level information encountered during training, which is lost during testing on a new corpus of logs.

VI. Conclusions

This paper explored several methods of improving the application of word embeddings for domain-specific sequence classification using a multi-input neural network. We conclude there are alternatives to word embeddings, achieving comparable performance, either increasing the granularity of the key-value extraction or replacing it using a hash-trick generation of vectors. However, these embedding alternatives rely more on lexical than semantic similarity. As such, there is a compromise between avoiding unknown words and extracting features that encode context. For domain-specific log message classification, if there is a training corpus enriched with a significant amount of domain-specific text, enrichment is the optimal approach.

We introduced increasing levels of specificity to our training corpora and quantified this with the coverage metric. Increasing the coverage beyond 50% does improve performance, but with diminishing returns. Fine-tuning with a corpus unique to a specific user can yield some benefit but could be marginal if the embedding already has a decent level of specificity.

We quantify a multi-input model's behavior by constructing training and testing environments with varying domain-specificity or coverage levels. We conclude that a low coverage environment model is less reliant on word-level information and is more consistent in correctly classifying log messages containing unknown words. A model trained in a high coverage environment relies more on word-level information and so will have consistently high performance on log messages with a high proportion of previously seen words. However, in a low coverage test environment, this model will be less consistent in classifying low coverage alerts than the low training coverage model. From these conclusions, we can train a generic model with an embedding enriched with as much domain-specific language as available at the time of training to produce good results across previously seen and newly introduced service environments. We can train a specific model trained with an embedding that includes language relevant to a particular set of service environments. The latter model can serve as a "worst-case scenario" model. It has some level of domain-specificity but will inevitably have low coverage for unseen service environments. It can leverage the character-level learning to classify with greater confidence.

We aim to investigate the state-of-the-art deep bidirectional embeddings and their efficacy for log message classification. We postulate that we can benefit from additional context when there is overlap in log messages and label definitions. We aim to test the "worst-case scenario" model on more unique service environments and gain a broader understanding of its adaptability to bespoke, unseen log messages.

References

[1] M. Ł. Steinder and A. S. Sethi, "A survey of fault localization techniques in computer networks," *Sci. Comput. Program.*, 2004.

[2] A. Dusia and A. S. Sethi, "Recent advances in fault Localization in computer networks," *IEEE Communications Surveys & Tutorials*, 2016.

[3] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," 2003.

[4] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013.

[5] H. Schwenk, "Continuous space language models," pp. 492–518, 2007.

[6] Y. Shehu and R. Harper, "Towards Improved Fault Localization using Transfer Learning and Language Modeling," *IEEE/IFIP Network Operations and Management Symposium*, 2020.

[7] A. Roy, Y. Park, and S. Pan, "Learning domain-specific word embeddings from sparse cybersecurity texts," 2017.

[8] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," 2018.

[9] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019.

[10] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," 2018.

[11] J. Lee, W. Yoon, S. Kim, D. Kim, S. Kim, C. H. So, and J. Kang, "Biobert: a pre-trained biomedical language representation model for biomedical text mining," *Bioinformatics*, Sep 2019. [Online]. Available: http://dx.doi.org/10.1093/bioinformatics/btz682

[12] D. Svenstrup, J. M. Hansen, and O. Winther, "Hash Embeddings for Efficient Word Representations," *NIPS 2017*, 2017.

[13] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," 2017.

[14] J. P. C. Chiu and E. Nichols, "Named entity recognition with bidirectional lstm-cnns," 2016.

[15] S. Merity *et al.*, "Pointer Sentinel Mixture Models," *Int. Conf. on Learning Representations*, 2016.

[16] "Stack Exchange XML Data Resource," 2019. [Online]. Available: https://ia600107.us.archive.org/27/items/stackexchange/

[17] Cisco, "Cisco IOS XR System Error Messages Reference Guide." [Online]. Available: https://www.cisco.com/c/en/us/td/docs/ios_xr_sw/error/message/ios-xr-sem-guide.html