IberSPEECH 2021
24-25 March 2021, Valladolid, Spain

An approach to intent detection and classification based on attentive recurrent neural networks

Fernando Fernández-Martínez[1], David Griol[2], Zoraida Callejas[2], Cristina Luna-Jiménez[1]

[1]Speech Technology Group, Center for Information Processing and Telecommunications, E.T.S.I. de Telecomunicación, Universidad Politécnica de Madrid, Spain
[2]Department of Languages and Computer Systems, University of Granada, Spain
fernando.fernandezm@upm.es, dgriol@ugr.es, zoraida@ugr.es, cristina.lunaj@upm.es

Abstract

Intent Detection is a key component of any task-oriented conversational system. To understand the user's current goal and provide the most adequate response, the system must leverage its intent detector to classify the user's utterance into one of several predefined classes (intents). This objective can also simplify the set of processes that a conversational system must complete by performing the natural language understanding and dialog management tasks into a single process conducted by the intent detector. This is particularly useful for systems oriented to FAQ services. In this paper we present a novel approach for intent detection and classification based on word-embeddings and recurrent neural networks. We have validated our approach with a selection of the corpus acquired with the Hispabot-Covid19 system obtaining satisfactory results.
Index Terms: topic classification, intent detection, conversational systems, recurrent networks, attentive rnn, attentive lstm

1. Introduction

Traditional spoken language understanding in conversational systems is divided into two main subtasks, intent detection and semantic slot filling, which are extended with domain recognition in multi-domain dialogue systems [1, 2].

The main objective of intent detection (also known as intent recognition or intent classification) is to classify user utterances into previously defined intent categories according to the domains and intents involved in user utterances [3, 4]. Users' intents can be defined as the will of the user (i.e., what they want to do). They are also denoted as dialogue acts, which can be defined as information actions that users share in the dialogue and are constantly updated (e.g., ask for train schedules, book a hotel, etc.).

This is a complex process that involves several key challenges: the lack of data sources to apply statistical methodologies (there are very few corpora with intents annotations and they are very difficult to obtain) [5], the irregularity of users' expressions (the use of colloquial expression, short sentences and broad content make very difficult to identify user's intents) [6], implicit intent detection (implicit intents are those in which users do not have clear intent requirements and it is necessary to infer the user's real intent by analyzing the set of possible intents defined for the task), and multiple intents detection (how to detect that the user's intent refers to more than one intent and correctly identify each one of them) [7, 8].

Intent recognition is a particularly useful task for conversational systems oriented to FAQ services [1]. In this kind of services, the intent recognizer can be used for both completing the natural language understanding task (by means of the detection of the main information pieces that are present in the user's utterances) and the dialogue management (by means of assigning the user utterance to one of the intents defined as possible responses for the system).

In this paper we present a novel approach for intent detection and classification based on word-embeddings and recurrent neural networks. We have validated our approach with a selection of the corpus acquired with the Hispabot-Covid19 conversational system, which was developed by the Spanish Goverment to provide responses to FAQ related to the pandemics originated by the Covid-19. The results of the evaluation shows the improvement of performance when using named-entities for intent recognition and embeddings adapted to the specific task, compared to our baseline approach (based on raw text with basic pre-processing).

2. Related work

As it has been described in the previous section, user intent detection plays a critical role in question-answering and dialogue systems. Traditional intent detection methods include rule-based template semantic recognition methods and method based on the use of statistical features, such as Naive Bayes, Adaboost, Support Vector Machines, and logistic regression [3, 4].

Current mainstream methods are mainly based on deep learning techniques and the use of word embeddings, which has been probed as a solution to better representational ability and domain extensibility instead of using bag of words [9].

Intent recognition methods based on deep learning techniques can be roughly classified into methods using convolution neural networks [10], recurrent neural networks [11] and their variants (LSTMs and GRUs) [7, 4], the Bidirectional Long short-term Memory (BLSTM) self-attention model [12], the capsule network model [13], the method of joint recognition [14], the use of distances to measure the text similarities (such as TF-IDF) [15], or methods combining several deep learning models [16, 17].

3. The Hispabot-Covid19 dataset

Hispabot-Covid19 is a conversational system developed for the Spanish Government to provide responses to frequently asked questions related to the pandemic originated by the Covid-19 and its implications in Spain. The system received more than 350,000 queries between April and June 2020. The assistant provided information from official sources, such as the Spanish Ministry of Health and the World Health Organization.

A total of 164 intents were defined for the system in order to classify the user's utterances and provide a response associated to each one of them. These intents are related to the symptoms

10.21437/IberSPEECH.2021-10

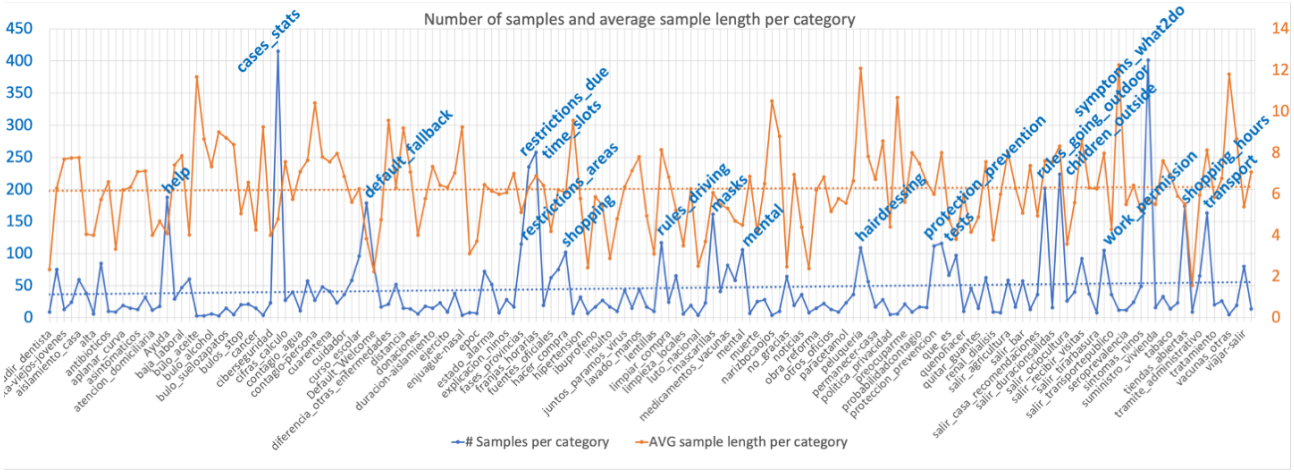

Figure 1: *Histogram of the amount of samples per class and average sample length per class*

of the disease, vulnerable groups, how it is transmitted, how to prevent and protect each self, living with infected people, conditions for isolation, official telephone numbers, among many others. The assistant also incorporated the information published in the Spanish B.O.E. regarding the application of the State of Alarm and the Plan for the Transition towards a New Normality. The assistant did not require or analyze personal data.

All interactions were recorded in the system and processed for continuous retraining and improvement of the chatbot. To this end, new training phrases and new question categories were incorporated daily, misunderstandings have been detected and corrected, and the knowledge base has been restructured according to updates in the information provided. Table 1 summarizes the main features of the selection of the corpus acquired with the Hispabot-Covid19 system that has been used to validate our proposal.

Figure 1 presents a histogram representation with the amount of samples available per each category (in blue) where most frequent categories (above 100 samples) have been highlighted in bold above the corresponding line. Besides, the figure also gives the average sample length per class (in orange). Mean values have also been computed and depicted for both data series (in dotted lines).

4. Model

As a solution for the sentiment analysis we have used a Recurrent Neural Network (RNN), a type of model widely used in the analysis of Twitter messages, for example. RNNs have the ability to process their inputs sequentially, performing the same operation, $h_t = f_W(x_t, h_{t-1})$, on each of the different elements that constitute our input sequence (i.e. words or, to be more exact, their corresponding embeddings), where h_t is the hidden state, t the time step, and W the weights of the network.

Table 1: *Main features of the selection of the Hispabot-Covid19 corpus used in the paper*

Param	Value
Number of samples	7532
Number of intents	164
Number of entities	55
Different words	7658
Average number of words per utterance	6.16
Average number of entities per utterance	0.6
Average number of words per entity	1.22

As it can be observed, the operation is formulated in such a way that the hidden state at each time step depends on the previous hidden states. Hence, the order of the elements in our sequences (i.e. the order of the words) is particularly important. As an immediate consequence, RNNs allow us to handle inputs (i.e. sentences) of variable length, which happens to be an essential feature given the nature of our problem.

Among the different possible architectures of this type of networks, we have opted for the so-called Long Short-Term Memory (LSTM) networks [2], a special type of RNNs that help preventing the typical vanishing gradient problem of standard RNNs by introducing a gating mechanism to ensure proper gradient flow through the network. LSTMs main characteristic is the ability to learn long-term dependencies. To do this, these networks are supported by basic constituent units called *cells* that are provided with mechanisms that allow deciding for each cell what information is preserved from that provided by the previous cells, and what information is provided to the next ones, both depending on the cell's current state.

4.1. Embeddings

(Word) embeddings are vector-type representations obtained for words in reduced-dimensional vector spaces where semantically similar words are always close to each other. The Fasttext project [18], recently open-sourced by Facebook Research, enables a fast and effective method to learn word embeddings that are very useful in text classification, clustering and information retrieval. In this work, the proposed model uses Fasttext word embeddings to represent the vectors for the words as input of the network.

At the time of training, FastText trains by sliding a window over the input text and either learning the target word from the remaining context (also known as continuous bag of words, CBOW), or all the context words from the target word ("Skip-gram"). Learning can be viewed as a series of updates to a neural network with two layers of weights and three layers of neurons, in which the outer layer has one neuron for each word

in the vocabulary and the hidden layer has as many neurons as there are dimensions in the embedding space. This approach is very similar to Word2Vec [19]. However, unlike Word2Vec, fastText might also learn vectors for sub-parts of words: so-called character n-grams. This ensures that for instance the words love, loved and beloved all have similar vector representations, even if they tend to show up in different contexts. This feature enhances learning on heavily inflected languages [20].

4.2. Model description

Our approach is based on a 2-layer Bidirectional-LSTM model with a deep self-attention mechanism which is represented in Figure 2. The model is implemented in Pytorch [21] and based on the architecture proposed in [22].

4.2.1. Embedding layer

The model is designed to work with sequences of words as inputs, thus allowing us to process any type of sentence. For this, a first embedding layer is provided that collects the embeddings $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N$ corresponding to each of the words $w_1, w_2, \ldots, w_N$ constituing the sentence we want to process, where N is the number of words in our sentence. We initialize the weights of the embedding layer with our pre-trained word embeddings.

4.2.2. Bi-LSTM layer

A standard LSTM model behaves in a unidirectional way, that is, the network takes as input the direct sequence of word embeddings and produces the outputs $\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_N$, where $\boldsymbol{h}_i$ is the hidden state of the LSTM cell at time step i, summarizing all the information that the network has accumulated from our sentence up to word w_i.

Instead, we have used a *bi-directional LSTM* (Bi-LSTM) that allows us to collect such information in both directions. In particular, a Bi-LSTM consists of 2 LSTMs, a *forward* $\overrightarrow{LSTM}$ that allows the analysis of the sentence from w_1 to w_N, and an *inverse or backward* $\overleftarrow{LSTM}$ which allows a similar analysis to be carried out but in the opposite direction, from w_N to w_1. To obtain the definitive outputs of our Bi-LSTM layer, we simply concatenate for each word the outputs obtained from the analysis performed in each specific direction (see Equation 1 in which $||$ corresponds to the concatenation operator and L to the size of each LSTM).

$$h_i = \overrightarrow{h_i} || \overleftarrow{h_i} \text{ , where } h_i \in R^{2L} \tag{1}$$

4.2.3. Attention layer

In order to identify the most informative words when determining the polarity of the sentence, the model uses a deep self-attention mechanism. Thus, actual importance and contribution of each word is estimated by means of a multilayer perceptron (MLP) composed of 2 layers with a non-linear activation function (*tanh*) similar to that proposed in [23].

The MLP learns the attention function g as a probability distribution on the hidden states h_i, that allows us to obtain the attention weights a_i that each word receives. As the output of the attention layer the model simply computes the convex combination r of the LSTM outputs h_i with weights a_i, where a convex combination is a linear combination of points where all the coefficients are non-negative and add up to 1.

4.2.4. Output layer

Finally, we use r as a feature vector which we feed to a final task-specific layer for classification. In particular, we use a fully-connected layer, followed by a *softmax* operation, which outputs the probability distribution over the classes.

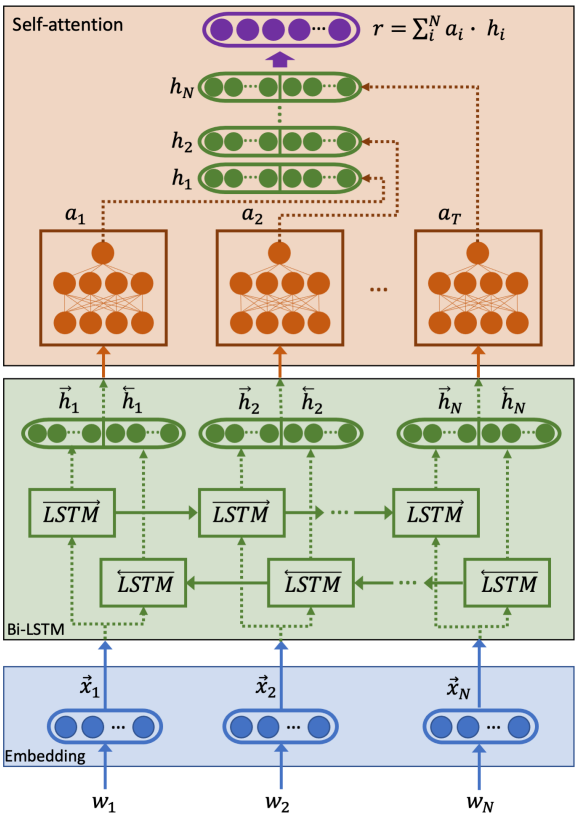

Figure 2: *Proposed model.*

5. Evaluation

According to the proposed methodology, we first pre-process the input text to transform it as a list of words. For this purpose, special characters are removed and numerical data are replaced by tokens, among other transformations. Subsequently, the remaining words are then mapped and transformed into their corresponding embeddings as the network can only work with numeric representations of each word. The type of word embeddings used in this work is described later in subsection 5.4. Specifically, the two different sets of word embeddings that have been applied are detailed in the next subsection. Once the input text is transformed into word embeddings by the Embeddings layer, the network is used to finally classify the text according to the identifed classes of the problem at hand.

5.1. Experimental setup

To prevent overfitting all the experiments have been carried out following a 5-fold cross-validation scheme. Each setup has been trained for 100 epochs and 'Stop-Early' has been adopted as the stopping criterion. Statistical tests for signifcance have been also applied to the results obtained in order to evaluate if there is a significant diference of performance depending on the adopted confguration.

Table 2: *Summary of results*

Model	Inputs	Embedding type	F1 score (%)
1	Pre-processed	General pre-trained	67.72
2	NER based	General pre-trained	72.58
3	NER based	Domain-specific	75.33

Table 3: *Length analysis*

	Total	Correct	Wrong
Number of samples	7532	5467	2065
Average length (in words)	6.16	5.87	6.93
Percentage	100	72.58	27.42

Although it could be possible to evaluate if there is any significant performance improvement when adjusting the hyperparameters of our model, in this work we did not aim at fine-tuning the network to achieve better performance than any other model in the state of the art, but rather to evaluate the sensitivity of our network in response to different pre-processing and embeddings choices. Another important aim has been to also develop a baseline for a deeper understanding of the dataset when addressing the problem of intent recognition.

The model was trained during 100 epochs using an Adam optimizer [24], with initial learning rate of 0.001, batch size of 32, and early-stopping after 5 epochs without improvement in the F1 classification score. Both bi-LSTM and attention layers had a 0.3 dropout rate. The encoder layers had a size L of 100. As a way to increase input variability from epoch to epoch, input embeddings are randomly added white noise with 0.15 probability rate in order to increase the robustness of the model.

As it can be deduced from Figure 1, some classes have more training examples than others. Hence, to prevent introducing bias in our models we apply class weights to the loss function, penalizing more the misclassification of under-represented classes. These weights are computed as the inverse frequencies of the classes in the training set.

5.2. Preprocessing of training data

Data preprocessing of our dataset goes through four main steps. These steps, described hereafter, are mainly intended to remove the noise present in the sentences and to encode the sentences in a way that can be used in the training of our network.

- All words are written in lowercase letters.
- Special words including emails, percentages, money, phone numbers, times, dates, urls and/or hashtags are assimilated and replaced by a special tokens, such as MAIL, DATE, URL,... to prevent information from being lost during data representation.
- The words contained in the datasets (or their corresponding entities) are replaced by their respective pre-trained vectors.
- The words, which are present in the datasets but not in the pre-trained word vectors, are considered Out Of Vocabulary (OOVs) words and simply discarded.

5.3. Named entity recognition

Named-entity recognition (NER) (also known as entity identification, entity chunking and entity extraction) is a subtask of information extraction that seeks to locate and classify elements in text into pre-defined categories such as the names of persons, organizations, locations, expressions of times, quantities, monetary values, percentages, etc. [25]. In this work, we have not implemented a Named Entity Recognizer but simply relied on manually annotated entities. All the sequences of words in the sentences of the dataset that are linked to relevant entities for the intent recognition task, such as dates, activities, symptoms, or covid and location names, have been carefully identified and annotated. Hence, the dataset comes with full annotation of entity-related items that will be useful for engineering feature extractors for Named Entity Recognition in the future.

5.4. Word embeddings

Using pre-trained word embeddings vectors as inputs of machine learning models is always a interesting option when training data corpora are restricted or limited. Hence, for our first approach we have used a set of pre-trained Fasttext vectors made up of a total of 2 million embeddings of dimension $W = 300$ generated for the Spanish language from different texts massively and automatically retrieved from the web and, in particular, from Wikipedia [26].

Alternatively, we have also used Fasttext to train task-specific embeddings from scratch. A setup similar to the one used for general pre-trained models was adopted. As a result, we also obtained CBOW models but with a smaller size: dimension was set to 100 in this case, consistently with the smaller size of our dataset.

5.5. Results

We have evaluated three different models whose main characteristics and results are detailed in Table 2. For the calculation of F1 we used the *weighted* version that takes into account the number of examples available for each different class.

The first model, that was adopted as our baseline, was directly trained from the available sentences after applying them the basic pre-processing described in section 5.2. General pre-trained embeddings were used for the tokenized words.

Compared to our baseline, the second model successfully introduced the use of the named-entities version of our dataset while the third one, the top performing model, also combined it together with task-specific embeddings trained on our dataset.

As it can be observed in Figure 1, there are some evident asymmetries in the distribution of the data. Therefore, we decided to further explore whether the sentence length affects the final performance on the assumption that shorter sentences are more difficult to classify. To that end, we computed the average length of correctly and incorrectly recognized sentences, respectively. However, the results obtained of this length analysis, summarized in Table 3 for the second model, suggested that the hypothesis did not hold for our data.

Finally, although omitted in this work, error analysis indicates that errors are mainly due to semantically overlapped categories, such as “shopping” and “opening hours” or “traveling” and “transports”, which suggests that a simplified and reduced set of intent categories may be also considered.

6. Conclusions

In this paper, the intent recognition results obtained in the evaluation of different pre-processing and embeddings configura-

tions of an attentive recurrent network are presented. The results indicate that there may be a large improvement of performance when using named-entities for intent recognition, relying on manual annotation, compared to our baseline approach (based on raw text with basic pre-processing), which suggests that conveniently parsed text should more accurately link entities to intents. A similar result may also be observed when relying on embeddings adapted to the specific task (i.e. trained on task-specific data), which suggests that, although general-purpose embeddings pre-trained on larger corpora reasonably fit the task, training corpus size is suitable for optimizing and building custom vectors for these specific domain and task. Finally, the Hispabot-Covid19 data set is a novel resource for researchers. This is the very first work addressing intent recognition based upon this dataset, thus all the experiments carried out in this paper should provide a relevant insight to those researchers.

7. Acknowledgements

The work leading to these results has been supported by the Spanish Ministry of Economy, Industry and Competitiveness through CAVIAR (MINECO, TEC2017-84593-C2-1-R) and AMIC (MINECO, TIN2017-85854-C4-4-R) projects (AEI/FEDER, UE).

8. References

[1] M. McTear, *Conversational AI. Dialogue systems, Conversational Agents, and Chatbots*. Morgan and Claypool Publishers, 2020.

[2] M. McTear, Z. Callejas, and D. Griol, *The Conversational Interface: Talking to Smart Devices*. Springer, 2016.

[3] J. Liu, Y. Li, and M. Lin, "Review of intent detection methods in the human-machine dialogue system," *Journal of Physics*, vol. 1267, 2019.

[4] S. Ravuri and A. Stoicke, "A comparative study of neural network models for lexical intent classification," in *Proc. of IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU-2015)*, 2015, pp. 702–709.

[5] S. Larson, A. Mahendran, J. J. Peper, C. Clarke, A. Lee, P. Hill, J. K. Kummerfeld, K. Leach, M. A. Laurenzano, L. Tang, and J. Mars, "An evaluation dataset for intent classification and out-of-scope prediction," in *Proc. of EMNLP-IJCNLP'19*, 2019.

[6] W. Xie, D. Gao, R. Ding, and T. Hao, "A feature-enriched method for user intent classification by leveraging semantic tag expansion," *LNAI*, vol. 11109, pp. 224–234.

[7] G. H. E. A. e. a. Firdaus, M., "A deep multi-task model for dialogue act classification, intent detection and slot filling," *Cognitive Computation*, 2020.

[8] U. Park, J.-S. Kang, V. Gonuguntla, K. C. Veluvolu, and M. Lee, "Human implicit intent recognition based on the phase synchrony of eeg signals," *Pattern Recognition Letters*, vol. 66, pp. 144–152.

[9] J. Kim, G. Tur, and A. C. et al., "Intent detection using semantically enriched word embeddings," in *Proc. of IEEE Workshop on Spoken Language Technology*, Brussels, Belgium, 2016, pp. 414–419.

[10] H. Hashemi, A. Asiaee, and R. Kraft, "Query intent detection using convolutional neural networks," in *Proc. of Int. Conference on Web Search and Data Mining, Workshop on Query Understanding*, 2016.

[11] A. Bhargava, A. Celikyilmaz, and D. H. et al., "Easy contextual intent prediction and slot detection," in *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2013, pp. 8337–8341.

[12] K. Sreelakshmi, P. Rafeeque, S. Sreetha, and E. Gayathri, "Deep bi-directional lstm network for query intent detection," in *Proc. of 8th International Conference on Advances in Computing Communications (ICACC-2018)*, 2018, pp. 8337–8341.

[13] C. Xia, C. Zhang, X. Yan, Y. Chang, and P. Yu, "Zero-shot user intent detection via capsule neural networks," in *Proc. of the 2018 Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium, 2018, pp. 3090–3099.

[14] B. Liu and I. Lane, "Attention-based recurrent neural network models for joint intent detection and slot filling," in *Proc. of 17th Annual Conference of the International Speech Communication Association*, 2016, pp. 685–689.

[15] R. Rojowiec, M. C. Fink, and B. Roth, "Intent recognition in doctor-patient interviews," in *Proc. of 12th Conference on Language Resources and Evaluation (LREC-2020)*, 2020, pp. 702–709.

[16] H. Yu, X. Feng, and L. L. et al., "Identification method of user's medical intent in chatting robot," *Journal of Computer Applications*, vol. 38, no. 8, pp. 2170–2174, 2020.

[17] C. Yang and C. Feng, "Multi-intent recognition model with combination of syntactic feature and convolution neural network," *Journal of Computer Applications*, vol. 38, no. 7, pp. 1839–1845, 2018.

[18] E. Grave, P. Bojanowski, P. Gupta, A. Joulin, and T. Mikolov, "Learning word vectors for 157 languages," in *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

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

[20] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017. [Online]. Available: https://www.aclweb.org/anthology/Q17-1010

[21] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *NIPS 2017 Workshop on Autodiff*, 2017. [Online]. Available: https://openreview.net/forum?id=BJJsrmfCZ

[22] C. Baziotis, A. Nikolaos, A. Chronopoulou, A. Kolovou, G. Paraskevopoulos, N. Ellinas, S. Narayanan, and A. Potamianos, "Ntua-slp at semeval-2018 task 1: Predicting affective content in tweets with deep attentive rnns and transfer learning," *Proceedings of The 12th International Workshop on Semantic Evaluation*, 2018. [Online]. Available: http://dx.doi.org/10.18653/v1/S18-1037

[23] J. Pavlopoulos, P. Malakasiotis, and I. Androutsopoulos, "Deep learning for user comment moderation," *Proceedings of the First Workshop on Abusive Language Online*, 2017. [Online]. Available: http://dx.doi.org/10.18653/v1/W17-3004

[24] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1412.6980

[25] U. Yasavur, J. Travieso, C. Lisetti, and N. Rishe, "Sentiment analysis using dependency trees and named-entities," in *FLAIRS Conference*, 2014.

[26] T. Mikolov, E. Grave, P. Bojanowski, C. Puhrsch, and A. Joulin, "Advances in pre-training distributed word representations," in *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.