Pre-print of full paper presented at CONVERSATIONS 2017 - an international workshop on chatbot research and design, November 22, in conjunction with the 4th International Conference on Internet Science, Thessaloniki, Greece. The final version of the paper will be published in the conference post-proceedings as part of Springer LNCS.

# Towards Open Domain Chatbots — A GRU Architecture for Data Driven Conversations

Åsmund Kamphaug[1], Ole-Christoffer Granmo[1], Morten Goodwin[1], and Vladimir I. Zadorozhny[2,1]

[1] Centre for Artificial Intelligence Research, University of Agder, Norway
[2] School of Computing and Information, University of Pittsburgh, USA

**Abstract** Understanding of textual content, such as topic and intent recognition, is a critical part of chatbots, allowing the chatbot to provide relevant responses. Although successful in several narrow domains, the potential diversity of content in broader and more open domains renders traditional pattern recognition techniques inaccurate. In this paper, we propose a novel deep learning architecture for content recognition that consists of multiple levels of gated recurrent units (GRUs). The architecture is designed to capture complex sentence structure at multiple levels of abstraction, seeking content recognition for very wide domains, through a distributed scalable representation of content. To evaluate our architecture, we have compiled 10 years of questions and answers from a youth information service, 200083 questions spanning a wide range of content, altogether 289 topics, involving law, health, and social issues. Despite the relatively open domain data set, our architecture is able to accurately categorize the 289 intents and topics. Indeed, it provides roughly an order of magnitude higher accuracy compared to content recognition techniques, such as SVM, Naive Bayes, random forest, and K-nearest neighbor, which all seem to fail on this challenging open domain dataset.

## 1 Introduction

This paper considers a novel approach to creating an intelligent chatbot that can assist with questions related to law, health, and social issues. In general, chatbot models can be represented using the diagram in Figure 1 [13]. The most straightforward approach is to build a retrieval-based chatbot in a closed domain (focused on a limited number of topics). Retrieval-based chatbots work on pre-defined responses and may use specification languages, such as Artificial Intelligence Markup Language [10], to manually define the pre-defined interaction patterns. Generative models, on the other hand, have the ability to generate new responses on the fly. The most challenging case is generative technique working in an open domain with multiple topics, and this is our focus in this paper.

It should be noted that there are many chatbot tools currently available on the market [14]. They range from simple toolkits that can be used by non-programmers to more sophisticated systems from companies, such as Microsoft [4], IBM [3], Facebook, Google [2], and Amazon [1]. Meanwhile, none of the existing platforms can be reliably used for complex conversations and critical actions (e.g., making payments). It is expected that

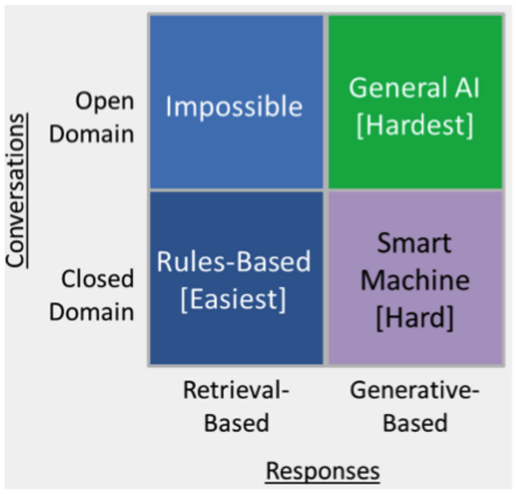


**Figure 1.** Taxonomy of chatbot models [13]

proper deployment of deep learning techniques will considerably improve this situation and our work is a step in that direction.

The paper has the following contributions:

- We propose an advanced neural network architecture (BRNN) for an intelligent open-domain chatbot (health, law, social issues).
- We collected large amount of the chatbot training data using efficient Web scraping.
- We provide a pre-trained TensorFlow based implementation of our model. In particular, our network is capable to accurately classify topic/intent for any given question.
- We implemented a chatbot system prototype and performed experimental study demonstrating that our approach outperforms top competitors.

The paper is organized as follows. The next section covers background and related work. We introduce our method in Section 3, and report on the experimental results in Section 4. Section 5 concludes and provides pointers to further work.

## 2 Background and Related Work

Several chatbot platforms are developed by tech giants including Api.ai (Google), Wit.ai (Facebook), LUIS (Microsoft), Watson (IBM), Lex (Amazon). All those bots differ in the way they model the conversation flow. For example, Api.ai first classifies the user request to determine if it matches a known intent and it uses a "Default Fallback Intent" to handle requests that do not match any user intent. Wit.ai should be taught by bot developers using examples. It processes similar user requests, extract major entities, and apply the logic defined by the developer. None of the commercial chatbot platforms utilize advanced deep learning methods, which make them considerably different from the approach that we propose in this paper.

Instead they apply alternative machine learning methods and related approaches that we compare to our method in order to stress its high efficiency. Decision Tree (DT) [11] is a non parametric supervised learning method used for classification and regression. This type of model aims to predict a target value by learning simple decision rules inferred from data features. The decision tree requires little data preparation, unlike many other techniques that rely on e.g. data normalization. One of the reasons that DT needs so little preparation is because it operates directly on both numerical and categorical data, taking advantage of decision thresholds. Another advantage of DTs is that they are easy to explain, facilitating translation to decision rules. DT learning can produce arbitrarily complex trees, however, data sparseness quickly becomes a problem as the tree grows, leading to over-fitting.

Random forest (RF) [5] is a meta-estimator that fits a number of decision tree classifiers on various sub-samples of the dataset. In all brevity, RF learning uses averaging to improve predictive accuracy and to combat the over-fitting of individual DTs.

The Naïve Bayes (NB) method is a supervised learning approach that is based on applying Bayes theorem with the 'naive' assumption that features are independent when the class is given. NB has performed quite well in many real world situations including document classification, intent recognition and spam filtering. In this paper, we explore two types of NB algorithms: Gaussian Naive Bayes (GNB) and Multinomial Naive Bayes (MNB). The GNB algorithm assumes that the distribution of continuous features is Gaussian. The MNB algorithm assumes a multinomial distribution and is often used in text classification. The input data for the latter algorithm is typical represented as word vector counts. In our experimental results, we report performance of these algorithms for both sequential and non-sequential word vector count data.

The K-Nearest Neighbors (KNN) method is a type of instance based lazy learning: KNN does not try to construct a general internal model, but rather stores instances of the training data. Then it simple uses majority voting applied to the nearest neighbors of the query point. Broadly stated, a query point is assigned the class that is most representative for the nearest neighbors of the point.

In contrast to the above discussed classifiers, Recurrent Neural Networks (RNNs) make inherently use of sequential information. In a traditional neural network we assume that all the input-output pairs are independent of each other. However, for many problem this assumption is not adequate. RNNs are called "recurrent" because they perform the same procedure for every element of a sequence, with the output at one step is transferred to the next. Another way to think about RNNs is that they have 'memory' that captures information about sequential events.

In [15] the authors consider how a chatbot can produce responses using deep learning and so-called LSTM cells in particular. They had access to 1 million twitter conversations from over 60 different brands. Their finding was that 40 % of the requests was emotional, and that their system was as good as a human agent at recognizing these emotions. Their study also showed that use of deep learning outperformed information retrieval systems, both from the perspective of qualitative human judgment as well as quantitative evaluation metrics.

In [9] the authors presented a solution that implements a novel chatbot system for a psychiatric counseling service. The system analyses the content of the conversation using neural natural language processing (NLP) techniques. It employs GRU-based sentence analysis [6,8] and the similarity of the sentences is estimated using the cosine similarity measure. Further, the system is capable of classifying 8 different emotions and extracting SNS dialogue. Finally, it is also able to track emotions over time based on a logging system that enhance data collection with meta data. Personal information like gender and age together with a knowledge base are used for generation of the personalized response. Note that similar approaches have been carried out for document classification using GRU based neural networks [16].

In [7], AIML (Artificial Intelligence Mark up Language) is used to create the FAQbot, a chatbot performing as an undergraduate advisor at the student information desk. This research uses a knowledge base that already contains the topics of all the available questions, which makes it different from our approach. The approach is further directly based on the chatbot system used for building the ALICE chatbot [12]. A major finding of this work is that domain-specific knowledge coupled with conversational knowledge yielded the best result, obtaining an accuracy of 21.1%.

## 3 Bidirectional Recurrent Neural Network for Intent Recognition

In this section we present our deep learning architecture for open domain chatbots, taking advantage of Gated Recurrent Units (GRUs) that are organized in a bidirectional fashion.

### 3.1 Neural Network Model

The neural network model is the central part of our approach. The overall model is shown in Figure 2 and explained below. In order to facilitate readability and implementation of our model, we provide a TensorFlow based implementation, with code pertinent excerpts interleaved in the text.

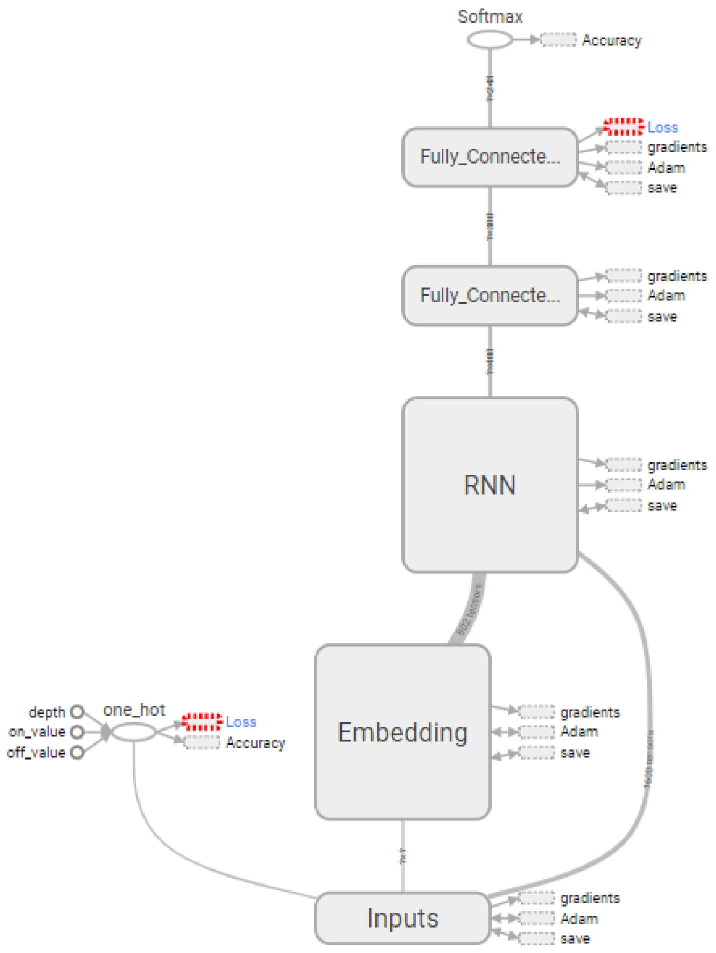


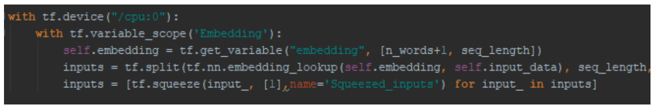
**Figure 2.** Our bidirectional GRU-based neural network model

The *inputs* to the network shown in the figure is the text to be analyzed, with the individual terms of the text being tokenized into numeric tokens, using one-hot encoding. That is, every question in the dataset is translated into a numerical representation.

An embedding layer is in turn used to create a vector representation of each term in vocabulary, resulting in terms of similar meaning having a similar placement in the produced vector space. This facilitates better generalization from training data since terms can be reused in more contexts based on their semantic similarity.

The semantic similarity of terms can be visualized in terms of a graph, with each node representing a term, and edges represent significant similarity (e.g., measured using cosine-similarity). The vector representations of the words are learned gradually as text is processed, leading to clustering of the terms that have similar meaning, while less similar terms are separated into distinct clusters.

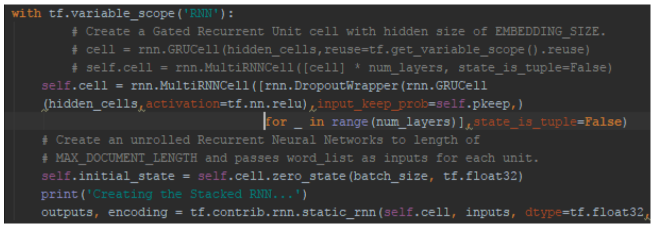
The input to the embedding layer is thus determined by the the size of the vocabulary and the length of the sentence, with the one-hot encoded terms the sentence being organized as a matrix. The code for the embedding layer is shown in Figure 3.



```
with tf.device("/cpu:0"):
    with tf.variable_scope('Embedding'):
        self.embedding = tf.get_variable("embedding", [n_words+1, seq_length])
        inputs = tf.split(tf.nn.embedding_lookup(self.embedding, self.input_data), seq_length,
        inputs = [tf.squeeze(input_, [1],name='Squeezed_inputs') for input_ in inputs]
```

**Figure 3.** The embedding layer code

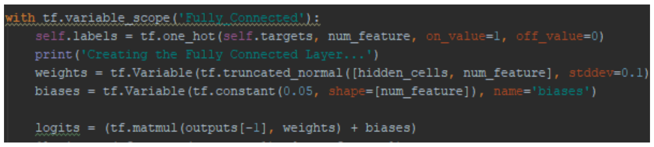
The next part of the network is the RNN. Here the learning of sentence interpretation takes place. The RNN is organized as multiple levels of so-called GRU cells [6,8], which is a variant of LSTM that only has two gates instead of three, thus simplifying the architecture. For the GRU cells we have used ReLU activation functions to combat the vanishing gradient problem. We also add a dropout layer to reduce over-fitting. Finally, a Multi-RNN cell stacks the existing cells upon each other to make another layer of abstraction, in order to learn more complex concepts. These RNN building blocks are used to create a Bidirectional Recurrent Neural Network (BRNN), which makes the network explore both the future and the past, before generating the output. Figure 4 shows our code for the RNN layer.



```
with tf.variable_scope('RNN'):
        # Create a Gated Recurrent Unit cell with hidden size of EMBEDDING_SIZE.
        # cell = rnn.GRUCell(hidden_cells,reuse=tf.get_variable_scope().reuse)
        # self.cell = rnn.MultiRNNCell([cell] * num_layers, state_is_tuple=False)
    self.cell = rnn.MultiRNNCell([rnn.DropoutWrapper(rnn.GRUCell
    (hidden_cells,activation=tf.nn.relu),input_keep_prob=self.pkeep,)
                              for _ in range(num_layers)],state_is_tuple=False)
    # Create an unrolled Recurrent Neural Networks to length of
    # MAX_DOCUMENT_LENGTH and passes word_list as inputs for each unit.
    self.initial_state = self.cell.zero_state(batch_size, tf.float32)
    print('Creating the Stacked RNN...')
    outputs, encoding = tf.contrib.rnn.static_rnn(self.cell, inputs, dtype=tf.float32,
```

**Figure 4.** The RNN layer code

The final part of our model includes two fully connected neural network layers, meaning that all the nodes are interconnected, ending up in a softmax-layer. Such a fully connected network is suitable for flexibly solving classification problems, after abstract features have been composed by the underlying layers. The code for the fully connected layers is shown in Figure 5.

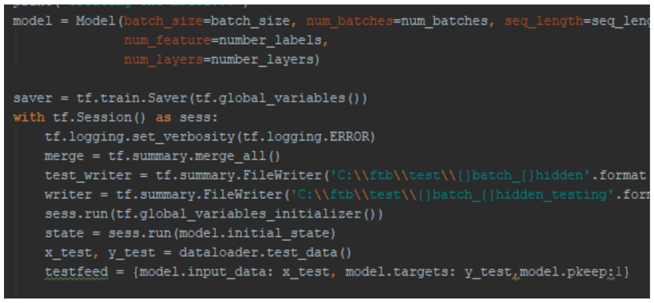


```
with tf.variable_scope('Fully Connected'):
    self.labels = tf.one_hot(self.targets, num_feature, on_value=1, off_value=0)
    print('Creating the Fully Connected Layer...')
    weights = tf.Variable(tf.truncated_normal([hidden_cells, num_feature], stddev=0.1)
    biases = tf.Variable(tf.constant(0.05, shape=[num_feature]), name='biases')

    logits = (tf.matmul(outputs[-1], weights) + biases)
```

**Figure 5.** The fully connected layer code

To interconnect the code fragments presented thus far, we now introduce the main toolbox that we have produced. All the configurations and parameters can be set here. We can also perform pre-processing, including vocabulary construction, tokenization of words, splitting the dataset into training and test sets, and creation of mini batches. Figure 6 shows an example session, where a complete neural network model is constructed. Finally, different test outputs are printed for the the user to monitor the network.



```
model = Model(batch_size=batch_size, num_batches=num_batches, seq_leng
              num_feature=number_labels,
              num_layers=number_layers)

saver = tf.train.Saver(tf.global_variables())
with tf.Session() as sess:
    tf.logging.set_verbosity(tf.logging.ERROR)
    merge = tf.summary.merge_all()
    test_writer = tf.summary.FileWriter('C:\\ftb\\test\\{}batch_{}hidden'.format
    writer = tf.summary.FileWriter('C:\\ftb\\test\\{}batch_{}hidden_testing'.for
    sess.run(tf.global_variables_initializer())
    state = sess.run(model.initial_state)
    x_test, y_test = dataloader.test_data()
    testfeed = {model.input_data: x_test, model.targets: y_test,model.pkeep:1}
```

**Figure 6.** Example on a session in Tensorflow

## 4 Experiments and Results

### 4.1 Experiment Setup

We have used a wide range of parameter settings and data set configurations to evaluate our approach. In this section, we report representative results.

### 4.2 Data Gathering & Preprocessing

We obtained our open domain data set from *ung.no*. This data set includes 200,083 data rows corresponding to different questions, answers and topics. Since our task is to classify the intent behind a question, we designed a query that could return questions, answers and topics. We here see a close resemblance between the intent of a question and the topic of the answer. We created a database class that used the *pymsql* package in Python to run the queries.

### 4.3 Experimental Results

We used the *ung.no* data with 200 083 questions and intents to conduct experiments with different machine learning algorithms. The algorithms were tested on sequential and non-sequential data. Table 1 reports the results obtained in this experiment.

All algorithms were trained for 10 epochs (the algorithms were allowed to pass over the dataset 10 times). After training the performance of each of the algorithms was validate using a validation data set that contained 10% of the total data. In this experiment, we also used the main topics provided in the train and test sets to measure performance. We observe, that our BRNN method significantly outperforms other algorithms. As shown in the Table 1 our method provides the highest accuracy of 70% on the validation set. SVM (RBF) is second best, having an accuracy of 40%.

**Table 1.** Accuracy of topic classification algorithms after 10 epochs using main topics

| BRNN | DT | RF | NB | MNB | KN | SVM(linear) | SVM (RBF) |
|---|---|---|---|---|---|---|---|
| 71.2% | 6.9% | 5.7% | 0.12% | 0.17% | 25% | 27% | 40% |

In the next experiment we also used the sequential data with training on 10 epochs. The results are reported in Table 2. In this experiment the overall accuracy is lower then in the previous experiments. This is because the model did not use the main topics (44), but rather works directly on the sub topics (289). We observe that a higher number of topics makes it harder for the model to classify the questions correctly. Meanwhile, our method also performs much better in the instance where the main topics are not used: BRRN has an accuracy of 31%, while the second best approach, RFs, achieves only 5.1% of accuracy.

**Table 2.** Accuracy of topic classification algorithms after 10 epochs using sub-topics

| BRNN | DT | RF | NB | MNB | KN | SVM(linear) | SVM (RBF) |
|---|---|---|---|---|---|---|---|
| 31% | 3.5% | 5.1% | 0.1% | 0.25% | 2% | 3% | 5% |

In the next experiment sequential information was not used. Instead we used bag of words, which is more favorable for other algorithms. We observe, that while the overall accuracy was much higher for all the other algorithm, our method still significantly outperforms all competitors (Table 3).

In the last experiment the algorithms were tested with the bag of words on the full range of sub topics. The results are in the Table 4. The BRNN demonstrated the accuracy of 31.3% and the second best Naive Bayes method has 10.3% of accuracy. This experiment also show that our BRNN steadily outperforms the competitors.

Figure 7 shows how the accuracy of the model for BRNN over time using the 44 main topics. This figure also shows the difference of the training data and validation

**Table 3.** Accuracy of topic classification algorithms after 10 epochs using main topics with Bag of Words

| BRNN | DT | RF | NB | MNB | KN | SVM(linear) | SVM (RBF) |
|---|---|---|---|---|---|---|---|
| 71.2% | 55% | 64.6% | 62.9% | 65.5% | 63.5% | 63.7% | 65.5% |

**Table 4.** Accuracy of topic classification algorithms after 10 epochs using sub topics and Bag of Words

| BRNN | DT | RF | NB | MNB | KN | SVM(linear) | SVM (RBF) |
|---|---|---|---|---|---|---|---|
| 31.3% | 5.6% | 5.4% | 10.3% | 7.7% | 4.9% | 7.6% | 4.6% |

data. The purple line is the training data and the blue is the validation data. As we observe in this figure, the accuracy for training data continue to climb, while the validation data accuracy is getting flat around 70 percent. This shows that the model is over-fit and it is possible to obtain better result if the model is more generalized. Since the accuracy for training data continue to increase, the model can potentially learn more from the training data, however, over-fitting must be fought.

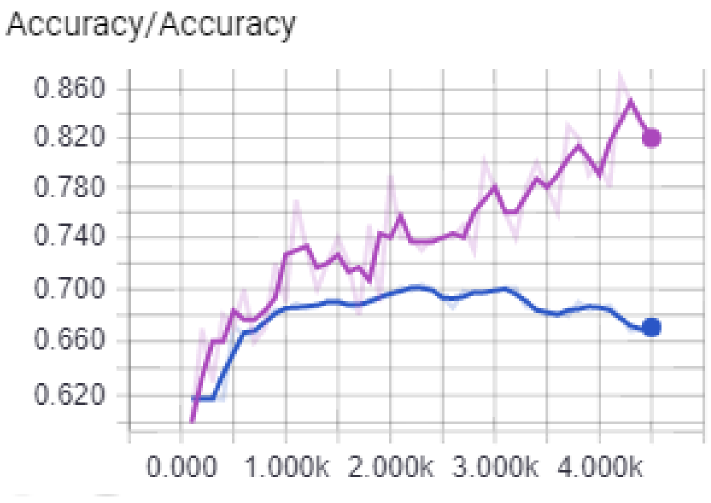


**Figure 7.** Accuracy for testing and training over 10 epochs using main topics

Figure 8 shows the accuracy of the model over time when for the 289 sub topics. This figure also shows the difference between the training data and validation data. The orange line corresponds to the training data and the teal line corresponds to the validation data. This figure shows almost the same result as in Figure 7, but the accuracy is lower overall, since classifying this many topic is much harder then the topic parents.

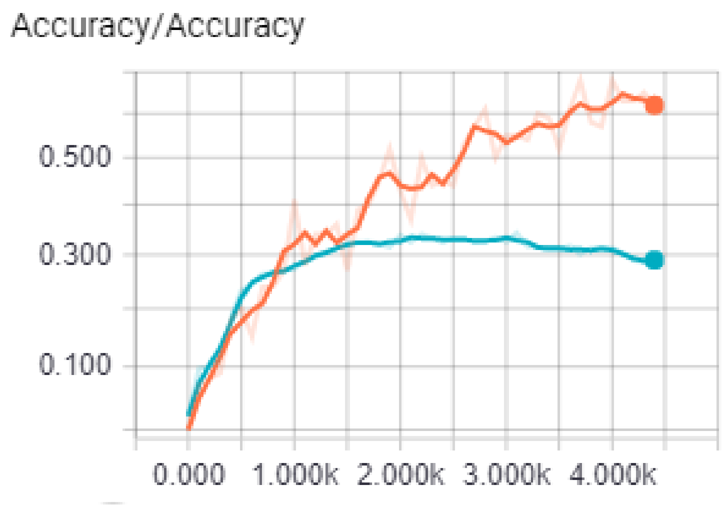


**Figure 8.** Accuracy for testing and training over 10 epochs using sub-topics

## 5 Discussion and Conclusion

The problem of building a generative open-domain chatbot for answering law, health, and social related queries is very hard. It requires accurate natural language understanding and comprehensive open-domain knowledge. In this paper, we proposed an advanced deep learning method based on recurrent neural networks, which considerably outperform related techniques. In particular, the experiments that have been reported in this paper show that a deep learning network or more specifically, a Bidirectional Recurrent Neural Network, performs surprisingly well on the task of recognizing intents and topics of open-domain text. We believe this can be explained by the capability of BRNNs to extract and remember abstract concepts occurring over sequences of words, simply by attempting to predict which word occurs next. Indeed, our BRNN outperforms all related algorithms, achieving an accuracy of 71.2%.

Our reported architecture and results is thus a significant step toward building an open domain chatbot, since our model is able to classify a wide range of topics from multiple areas, such as health, law, as well as social and personal issues. Despite the openness of the domain, our architecture was able to fetch answers based on topics quite accurately.

In the future, we intend to investigate how context can be used to guide generation of conversation, wiring context directly into the neural network architecture. We also intent to enhance our BRNN using more general bodies of text on various topics, to facilitate transfer learning to the chatbot domain.

## References

[1] Amazon Lex, `http://docs.aws.amazon.com/lex/latest/dg/what-is.html`

[2] Api.ai, `https://api.ai/`

[3] IBM Watson Conversation Service, `https://www.ibm.com/watson/developercloud/conversation.html`

[4] Microsoft Bot Framework, `https://docs.botframework.com/en-us/`

[5] Breiman, L.: Random forest. Machine Learning (1999)

[6] Chung, J., Gulcehre, C., Cho, K., Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555 (2014)

[7] Ghose, S., Barua, J.J.: Toward the implementation of a topic specific dialogue based natural language chatbot as an undergraduate advisor. In: 2013 International Conference on Informatics, Electronics and Vision, ICIEV 2013 (2013)

[8] Greff, K., Srivastava, R.K., Koutník, J., Steunebrink, B.R., Schmidhuber, J.: Lstm: A search space odyssey. IEEE transactions on neural networks and learning systems (2017)

[9] Lee, D., Oh, K.J., Choi, H.J.: The chatbot feels you - A counseling service using emotional response generation. In: 2017 IEEE International Conference on Big Data and Smart Computing, BigComp 2017 (2017)

[10] Marietto, M.d.G.B., Aguiar, R.V., Barbosa, G.d.O., Botelho, W.T., Pimentel, E., Franca, R.d.S., Silva, V.L.d.: Artificial Intelligence Markup Language: A Brief Tutorial. International Journal of Computer Science and Engineering Survey 4(3), 1–20 (2013)

[11] Rokach, L., Maimom, O.: Data mining with decision trees: theory and applications (2014)

[12] Shawar, B.A., Atwell, E.: ALICE chatbot: Trials and outputs. Computacion y Sistemas (2015)

[13] Stefan Kojouharov: Ultimate Guide to Leveraging NLP &amp; Machine Learning for your Chatbot, `https://chatbotslife.com/ultimate-guide-to-leveraging-nlp-machine-learning-for-you-chatbot-531ff2dd8`

[14] Walker, J.: Chatbot Comparison – Facebook, Microsoft, Amazon, and Google (2017), `https://www.techemergence.com/chatbot-comparison-facebook-microsoft-amazon-google/`

[15] Xu, A., Liu, Z., Guo, Y., Sinha, V., Akkiraju, R.: A New Chatbot for Customer Service on Social Media. Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (2017)

[16] Yang, Z., Yang, D., Dyer, C., He, X., Smola, A.J., Hovy, E.H.: Hierarchical attention networks for document classification. In: HLT-NAACL. pp. 1480–1489 (2016)