An Ensemble Approach to Artificial Conversations

Kavin Chandrasekaran
kchandrasekaran@wpi.edu

Abstract:

In an open-domain conversational scenario with an AI system, the AI should be able to respond in a way that is meaningful and appropriate. Generating responses in a conversation involves natural language understanding and the natural language generation. The main aim of this project is to create an algorithm that improves the natural language generation process based on a sense of emotional awareness derived from conversation history. The proposed approach is primarily focused on the emotional influence part in the artificial conversation. The proposed approach is based on an ensemble technique using multiple neural networks and trees.

Introduction and background:

With the advent of many personal assistants like Cortana, Siri, Alexa and Google Assistant, people are getting used to the concept of having a conversation with an AI system. While the personal assistants are good in closed-domain scenarios, in which they are already trained to answer, they are not good in having an open-ended conversation. The major driving factor in determining the path of an open-domain conversation is the response a person gets during each exchange. For example, if a person is having a strong opinion about something, how would an artificial intelligence system respond to it? Does it make a contradicting statement and start an argument or does it agree with the person or does it start replying with insults [1] to undermine the conversation? This decision could be made based on the emotional intelligence. According to Salovey and Mayer [2], emotions can be considered as a response to certain events and can be manipulated to control future action. We can consider a conversation as an event that could be used to influence the outcome of a relatable future action by crafting the responses in a specific way [3]. Researchers are working on improving how to improve the quality of the responses from an AI, by adding more features like context awareness and personality [4, 5]. The proposed work will try to use the emotion as a feature to improve the quality of the response. The emotion is based on learning from the conversation history how the person responds for each kind of reply. The algorithm would make a decision on how to reply based on a utility function. The best part about an interacting system is that it can learn new things as it goes on by just asking for a label when it encounters something it doesn't know. In the proposed algorithm, I will be using a combination of neural networks, clustering and decision trees. A neural network is collection of neurons that are designed to output values that would represent if the neuron was triggered based on the weights and thresholds for the neuron and the input value. The neurons can be perceptrons, which have binary output or it could be sigmoid neurons which outputs a continuous value. I will be using multiple layered sigmoid neuron based neural network known as Multilayer perceptron network (MLP). Fig 1 shows the structure of a MLP network. There are many methods to create a context aware conversation system using neural networks [6]. There is a sequence to sequence method which uses a sequence of inputs to identify the contexts and generate a sequence of responses [7, 8]. An ensemble approach is to use a combination of machine learning methods to make the algorithms perform better

Methodology:

The proposed algorithm has four major segments, understanding the input,

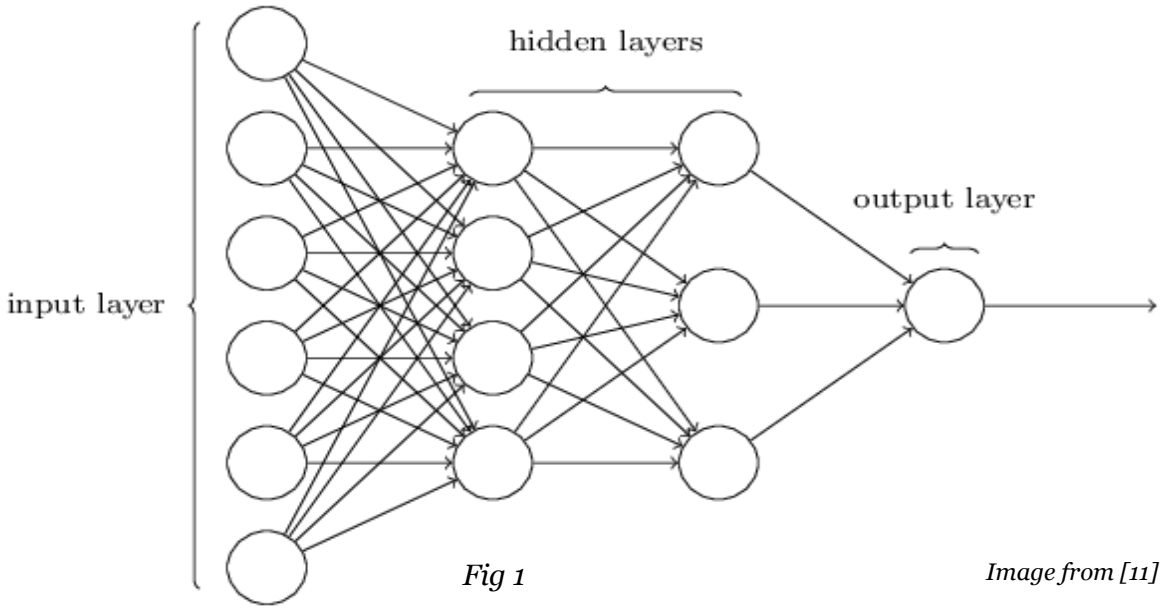


Fig 1 *Image from [11]*

identifying emotions, making decision on how to respond and creating a response. The algorithm would the use of an ensemble approach for doing all the above mentioned segments. A high level overview of a single step of the algorithm is shown in fig. 2. From the proposal, there were a few changes to the algorithm, which had to be made because of the temporal and computational constraints. I haven't had a chance to find out what type of a sentence the input is. This could have been helpful in the performance of the reply generation part as we will see later.

The input from a person is a sentence for a conversation. The algorithm first uses a neural network that does part-of-speech tagging [10]. The neural network will be trained to identify the part-of-speech based on a probabilistic model. Originally in the proposal, the idea was to do the tagging first and see how well it performs and how we could use the tagged data as input for the subsequent methods. But, later I realized that the part of speech tagging was not needed till the end. So, these set of tagged words are sent to the natural language generation system

The input words are then taken by another neural network that determines if and what type of emotion are represented by the words. Contrary to the proposal, I have changed the approach from complete unsupervised approach to a supervised learning method with classification. The primary reason is because of the availability of labelled data and it could be a good start to have a few emotions that are identifiable and then improve learning new emotions.

The neural network does a classification with room for unobserved classes. In the future, maybe we can expand on that, and find emotions based on word patterns and use those in the scale for clustering. In future, clustering could be done using DBSCAN and the cluster ID can be used to identify them.

Since we have distinct emotions from the previous analysis, I decided to use trees to store and track them. This is to record how the emotions change during the conversation and if we have a utility function that is set to have target emotion of *Happy*, then upon evaluating using emotional analyzer, it should be happy. The utility function is an identifier to represent what is the desired emotion state to drive the conversation. If the utility function is set to happy, and happiness is an identified emotion state, then the trees which end in that state are selected. The training data will have conversation transcripts and based on how each conversation is proceeding, decision trees will be created. If the emotion doesn't fall under any existing clusters, a tree would be created and the emotions of the previous 5 states and added as its ancestors. During

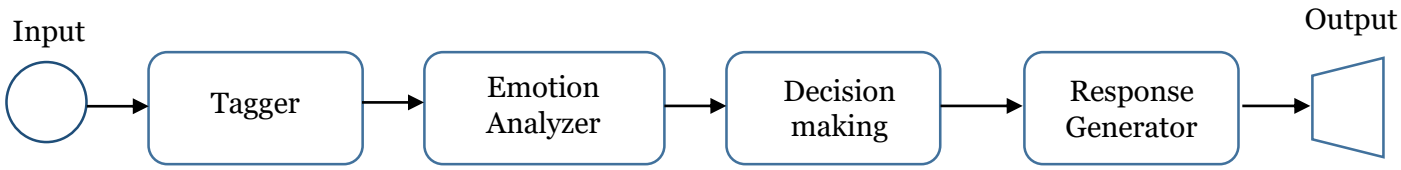


Fig 2

the following iteration of the conversation the decision tree will be updated upon the emotion of the response. This way, the system will be constantly verifying and updating the trees based on the conversation history.

Based on the selected path from the trees, the natural language generation system would create the response. The NLG system will first analyze the input in sequences, taking into consideration not just the current inputs but also previous transaction. Then it generates the response with probabilistic model which was trained using conversation transcript datasets. The response generated will be the output from the algorithm and would complete a conversation.

Approach:

To begin our approach, we start by doing some preprocessing in the data, like cleaning the missing and unnecessary values and reshaping for it to be used in the model.

First, we will talk about the ***Part-of-Speech tagger***. The system is created on an LSTM RNN. LSTM is special kind of neural network which has memory associated with it. That's why it is called as the Long Short Term Memory. With the help of the memory the neural net is not only available to the data in the current state, but also in the previous state. The training is done for the tagger which will be trained using the Brown Corpus [12] dataset which has a collection of texts which has the part-of-speech tagged words. The tagger was trained using two different machine learning model to have a comparison of the performance. I tried tagging with LSTM NN and also with a random forest. A random forest is good in classification, since it has so many trees each having different split functions and finally conglomerating them and picking which was voted the most. The bootstrap resampling method was included in random forest. The evaluation criteria will be the accuracy with which the tagger is able to identify the tags. Once our classification models are tuned and evaluated, we move on to the next step.

	word	pos_tag
0	the	AT
1	Fulton	NP
2	County	NN
3	Grand	JJ
4	Jury	NN

The next step is to do the ***emotional analysis***. This is done by training a simple feedforward multilayer perceptron network. For this analysis we found a few datasets which contain data from social network like tweets and facebook posts and they were tagged with labels for different emotions like Sad, Angry, Fear, Love, etc., 13 in total. Those were all the labelled data, but not all the data will be labelled. So we wanted to use a probabilistic model which could find the probabilities of each label given. This is helpful because, in my setup, I used one additional class to map anything that didn't fall under the existing categories.

Now about the actual machine learning model itself, I used, a simple feedforward propagation model. The feed forward propagation was a simple mulit-layer perceptron, which has a sigmoid activation function. Since the output are straight forward, I thought this could be good enough. Also, this game an opportunity to explore different neural network models. The back propagation was taking a lot of

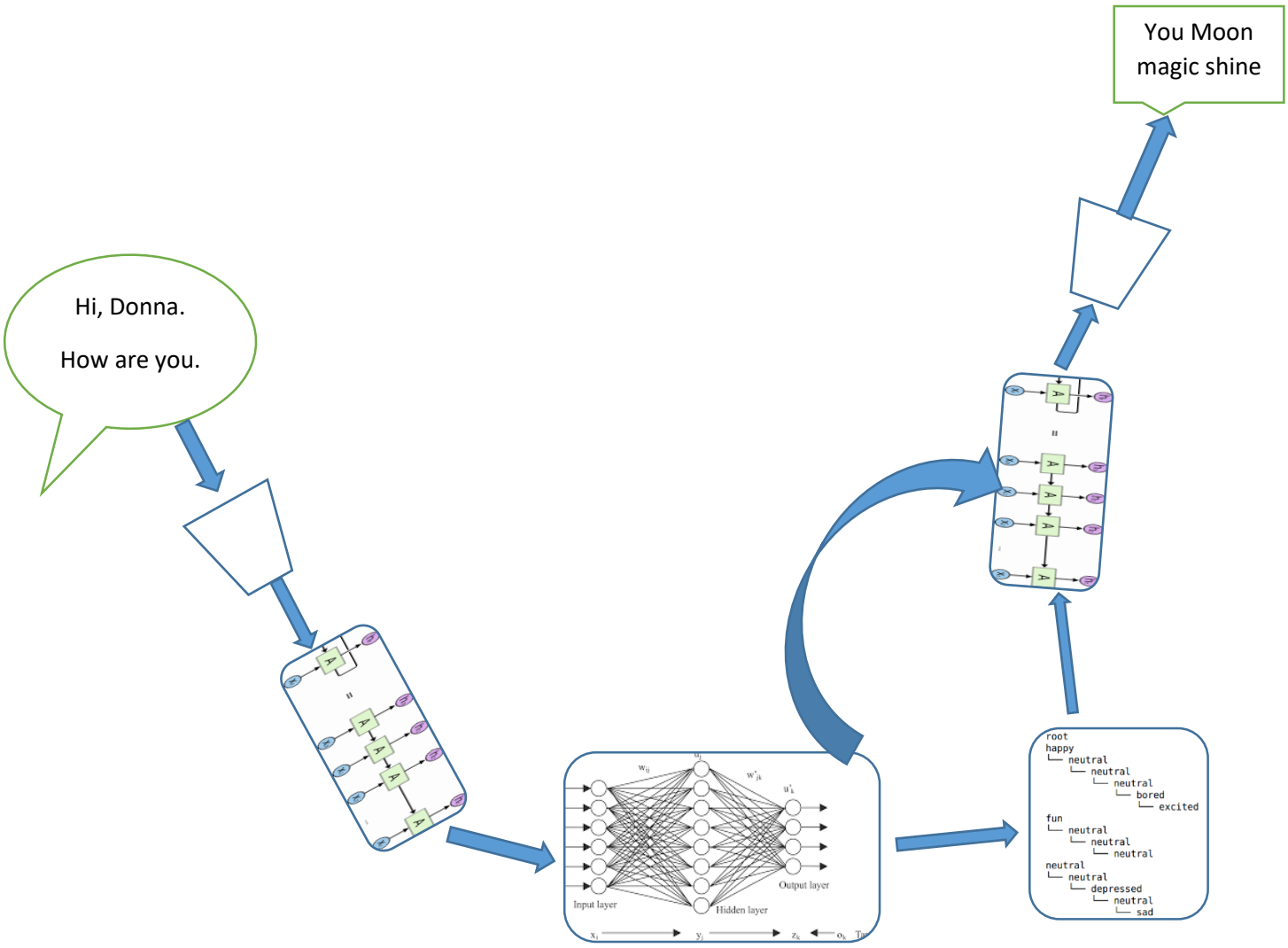
computational resource but it was learning as epochs passed by. But the multilayer perceptron did not learn from its mistakes and hence stayed constant. But different configuration of the network would give different performance and I was able find a configuration based on trial and error which gave me good accuracy.

After the emotional analysis, we have an emotional state that is represented as a node in a tree. ***Trees*** in this case used as a way to track the state transitions between those emotion state transitions. Let's say there system is new and have no recorded nodes of emotion. Then the first recorded emotion is stored as the root of the tree, and the consecutive emotions are added as the children. In this method, the AI agent has a custom built forest, which can store a collection of trees and can search for a node in the entire forest and there is a path found, it will isolate the tree and extract the path. This custom process will help in locating previous emotions and what is the shortest way to reach the desired emotion.

Each AI Agent is assigned with a utility function. The utility function is the method that decides what the AI is working towards. Since we are doing emotional manipulation in artificial conversation, our goal is to make the other person reach a specific emotional state. The utility function of our AI agents will usually have a specific emotion that they know about and they are working towards. At this point the utility function is tied to a AI agent, in the future it could be dynamic, which could be controlled by the human or the AI itself.

Once the emotion state the AI is manipulating towards the utility, we get to the next interesting part of the ***text generation***. In the text generation too I am using the LSTM neural network. The LSTM is a flavor of recursive neural network and which means it doesn't just look at the input value but also looks at the previous inputs as well. This can be used to create a probabilistic model that can take continuous input and provide continuous output with context of what was produced before. This is perfect example for the using natural language processing. The texts are nothing but a series of continuous characters. We can use a probabilistic model to find the probability of the next word or letter. For this we need to first the list of all the words that are used in the test. This is tricky since, there are so many words which are spelt differently in different places, typos, shortening, etc. So in any written data set we have to clean for those unwanted entities in our dataset. For training the model in my algorithm, I used a few different datasets, one was a movie dialogue corpus and the other was twitter interaction dataset. The twitter interation data set had emotional tags as well, so I used that in training the neural network for emotional classification too.

Once the data was cleaned, I used a bag of words method, which lets us convert words into unique numerical value, so that they can be used in mathematical calculations. Though there are other sophisticated methods like n-grams, feature hashing, I thought I could start simple and if it doesn't work, I could explore the complicated ones. Moving onto the neural network, the input layer had all the possible input values and only one LSTM layer. Though adding one more layer seemed like an enticing option to improve the performance, in the brief amount of time I tried the two layers the memory was filling up so fast, I had to stop it from crashing the machine. So the text generation neural net was trained based on the conversation. To get that I used the movie corpus dataset, which had a complete transcript of movies. From that, I extracted only the conversations, and split them in to input and output. Understanding that this is based on perspective and one person's response is other person's input, I combined the data to get the input and output values so that when the input values are read the corresponding response from the person will be output. In these split data sets, the model will be trained on.

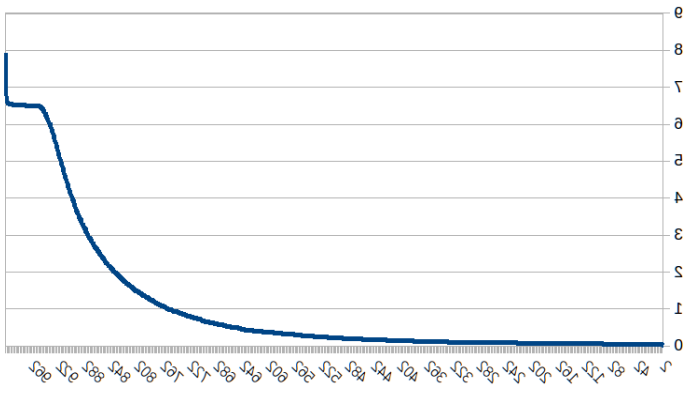


The below fig shows that overall structure of the setup and in terms of hardware, the entire setup was running on Intel x99 14K series processor, with 16 GB of RAM and 2TB of storage. The Graphics card was not compatible to be used with TensorFlow, so used only the CPUs and its overclocking features.

Experiments and evaluations:

Initially, I built and trained the components individually and I was testing different methods for each component. Starting with the tagger, I tried different levels of LSTM layers but it didn't improve the performance, so refrained from doing that. I performed a 80/20 split for training and testing datasets and did the predictions for the POS classification. The confusion matrix shows that the model did better in the most commonly occurring Parts of speech but there were many POS tags which were sparsely populated , so the total accuracy went down to about 13%. The reason I picked LSTM for the tagging was, that what a word could represent is best known by looking at the previous words. That's the basic idea of the auto complete in cell phone text and automatic text generation. Predicting the next part of speech is a precursor to predicting what word would actually be used. After training a few times, I saved the model and the weights which gave the least loss. The loss function is the inverse log function and we are just trying to minimize it.

$$\text{loss} = -\frac{1}{N}\sum_{i=1}^{N} \ln p_{\text{target}_i}$$



The above inverted graph shows the loss value reducing as the epoch passes by. The below table shows that the tagger performs better when there are more instances of the class. The data is highly unbalanced and maybe the accuracy would increase after handling the unbalanced-ness by under-sampling or any other methods.

Since the LSTM RNN didn't give so much aaccuracy, I tried to use random forest to see if it performed any better in the same problem.

preds actual	0	1	2	3	4	5	6	7	8	9	...
0	22232	3	0	0	0	0	0	0	0	0	...
1	0	8773	168	111	0	0	0	0	0	0	...
2	0	2573	29539	541	49	0	11	0	44	0	...
3	0	1289	220	10894	6	22	13	0	8	0	...
4	0	747	69	8	8887	0	0	0	0	0	...
5	0	1	150	21	0	238	0	0	0	0	...
6	0	12	10	7	0	0	21561	0	165	0	...
7	0	599	2	0	0	0	0	162	0	0	...
8	0	11	0	0	0	0	375	0	4657	0	...
9	0	0	0	0	0	0	0	0	0	615	...
10	0	1050	41	13	1	0	0	0	0	0	...
11	0	0	0	0	0	0	0	0	0	0	...

Once the tagger is set up, I started the neural network for classifying the emotions. My initial idea was so look at the specific combination of words and then do a clustering algorithm. That way after clustering we would know there are different emotions, but we could ask the speaker what they were feeling at that time. But since I found a labelled data, I thought it would be a good starting point, and then while the algorithm continues to learn, we can add one more component to improve the performance.

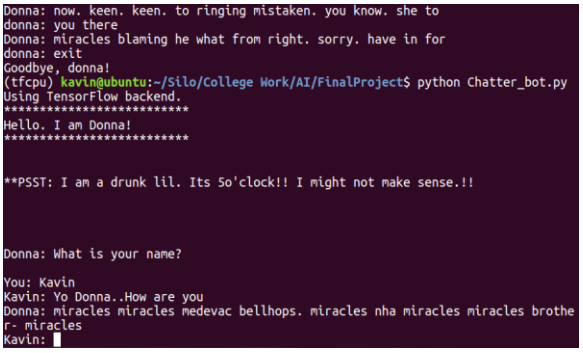
The emotional analyzer I tried backpropagtion NN and MLP with ReLU and sigmoid function. Backpropagation consumed too much resources and didn't finish. But the simple MLP had a testing accuracy of about 83%. There were a total of 13 classes and I added some more classes just so we can have some class to assign to, the values that don't fall under the predefined classes. This could be basis for the future expansion of the project with arbitrary emotion recognition.

Once the emotional analyzer is tuned, the emotional states have to be stored somewhere. So I custom created a simple forest where trees could be added and trees could be searched for nodes, roots, etc. That gave a comprehensive look into what was being stored in the trees. The trees didn't have any part that need to be evaluated. Information storage and retrieval in my custom methods were verified. One of the major roles of the tree is to find if there is a path between the current path and the utility function of the AI agent. Once a path or multiple path are identified, the shortest path will have to be chosen. Then the responses generated by the text generator must be skewed to match the emotion suggested by the tree path. The above is the sample of the tree structure in the forest.

```
root
happy
└── neutral
    └── neutral
        └── neutral
            └── bored
                └── excited
fun
└── neutral
    └── neutral
        └── neutral
neutral
└── neutral
    └── depressed
        └── neutral
```

Once the forest and tree infrastructure is setup, the final text generation. The text generation was trained on conversation dataset, like movie dialogues social media interactions. There was not really good way to evaluate the performance of the text generator. If we give a set of words from a sentence and if the generator also always produces the same word, then that would be overfitting. We are trying to train the model to identify the potential in that situation and make the best choice. But if we are looking for a specific value for a specific input, then we would be over fitting. To avoid unintentionally doing the overfitting, the dataset was split into different folds and training was done one set and the testing was done in another set. Since we had many folds, we just picked at random.

But still there is the problem about not being able to quantifiably determine if the model is good. The best I thought of was to run it and see if the output makes sense, which it didn't.



Since we were trying to make those emotion states happen in that order, it is important for the responses to have that emotion too. So if there is a text created for that response, it has to be run through the emotional analyzer and then make sure it is of the desired emotion. If not, the text generator would be asked to generate another text as a response. This didn't seem to have any effect, because every time, it was similar to the one before applying that condition. Of course, not the same text, but similarly random and vague and in comprehensible. Though my attempt to create a working emotionally aware AI system failed. I have learned a lot during this process and have many ideas for the future work.

Conclusion.

Artificial intelligence is a ubiquitous and helps in so many quantifiable field. But the psychological aspects are important as the physical aspects. An emotionally aware AI could help a lot in healthcare, hospice care, child care, etc. There are few areas people don't think, when they say they want to expand the reach of AI. Emotional and psychological support is important to every functioning person. If there is no person around to provide them, but if a computer could temporarily help, then it worth the time in finding a way to make it happen.

In this project, there were many important lessons learnt. Many of which are easily mitigatable during the next time. I have gained a greater understanding of the concepts and various techniques. Though this work fell short of the expectation, there are so many opportunities for improvement.

Future Work

One of the main reasons the responses generated by the AI was uncomprehensible, was because it was trying to do the predictions for inidvidual words. If the number of features it is tyying to predict and then use them to make another prediction is lesser, then the overall output would be better.

To achieve that, we can try an N-gram model where words are grouped in groups of N and then the prediction is not for 3 indivdual words but for one group. This reduces the number of predictions. But this does involve some additional preprocessing steps which would include finiding meaningful n-grams. Other methods we could use are called feature hashing, were features are grouped and hashed and the prediction is for the hash and not for all features.

References:

[1] http://venturebeat.com/2016/08/31/microsoft-chatbot-is-insulting-people-again-and-thats-a-good-thing/

[2] Salovey, Peter, and John D. Mayer. "Emotional intelligence." *Imagination, cognition and personality* 9.3 (1990): 185-211.

[3] Tan, Chenhao, et al. "Winning arguments: Interaction dynamics and persuasion strategies in good-faith online discussions." *Proceedings of the 25th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2016.

[4] Sordoni, Alessandro, et al. "A neural network approach to context-sensitive generation of conversational responses." *arXiv preprint arXiv:1506.06714* (2015).

[5] Yao, Kaisheng, Geoffrey Zweig, and Baolin Peng. "Attention with intention for a neural network conversation model." *arXiv preprint arXiv:1510.08565* (2015).

[6] Shang, Lifeng, Zhengdong Lu, and Hang Li. "Neural responding machine for short-text conversation." *arXiv preprint arXiv:1503.02364* (2015).

[7] Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le. "Sequence to sequence learning with neural networks." *Advances in neural information processing systems*. 2014.

[8] Vinyals, Oriol, and Quoc Le. "A neural conversational model." *arXiv preprint arXiv:1506.05869* (2015).

[9] Serban, Iulian V., et al. "Building end-to-end dialogue systems using generative hierarchical neural network models." *arXiv preprint arXiv:1507.04808* (2015).

[10] Schmid, Helmut. "Part-of-speech tagging with neural networks." *Proceedings of the 15th conference on Computational linguistics-Volume 1*. Association for Computational Linguistics, 1994.

[11] http://neuralnetworksanddeeplearning.com/

[12]http://www.essex.ac.uk/linguistics/external/clmt/w3c/corpus_ling/content/corpora/list/private/brown/brown.html

[13] https://www.microsoft.com/en-us/download/confirmation.aspx?id=52375

[14] https://people.mpi-sws.org/~cristian/Cornell_Movie-Dialogs_Corpus.html

[15] https://chenhaot.com/data/cmv/cmv.tar.bz2