Automatic IDN Dialog Act Tagging of Design-Team Conversations

Homero Roman Roman
Stanford University
homero@stanford.edu

Abstract

In this paper we explore the effectiveness of using neural networks for the classification of conversations into their respective dialog acts. For this task, we use as our tags the Interaction Dynamics Notation (IDN) developed at the Stanford Center for Design Research. This notation is a set of 12 symbols that represent group interactions and depend on social context. Due to this context-dependency, tagging must not only take into account each individual's input but also the reactions of other members within the group. By using recurrent neural networks the hope was to capture this context and improve classification accuracy. Our research therefore shows that recurrent neural networks are an effective tool in automatic tagging of the context-dependent IDN notation for group interactions.

1. Introduction

Designing as a team tends to be more effective than as an individual because it leads to better decision making thanks to a wider access to information and a more critical evaluation of ideas. Due to the increased benefits, it comes as no surprise that companies are now pushing towards team-based design in the face of increased product complexity and as a way to push for shorter product development cycles (Kleinsmann & Valkenburg, 2005; Valkenburg, 2000).[6] However, group interactions are complex and cannot be solely explained as the sum of the contributions of its individuals. This is even more prevalent during brainstorming sessions where ideas flow freely from person to person and it is nearly impossible to track down their origin. In such situations, a person who does not seem to verbally contribute much to the conversation may be fiercely analyzing the ideas of everybody in her mind and suddenly come up with an insight the entire group overlooked because it was fixated on a specific problem formulation. Despite the complexity, these interactions are not necessarily caused by random strokes of inspiration. Instead, previous research in the field of team dynamics has theorized that there are common patterns within effective teams that are more conductive to effective problem-solving (Sonalkar et al).[7] This is true even across teams with different people and tasks. This indicates that group interactions may have a larger role than individual differences in leading to effective solutions. Even more exciting is the possibility that by diagnosing these patterns within a team one could coach a team to be more effective. Currently, management consulting services exist to coach management teams into being better at managing people for a certain project. However, there hardly exists a system for coaching design teams into making product breakthroughs or improving upon pre-existing products. Therefore, the vision of this project is helping in the construction of a real-time system that will be able to give feedback to teams as they are coming up with ideas and lead them to successful insights.

With this end goal in mind, the Interaction Dynamics Notation (IDN) was developed to provide a visual representation of the flow of conversations similar to the way music is represented in a staff. The IDN is a set of symbols "designed to represent the moment-to-moment flow of interpersonal responses in conversational interactions in design teams." Speaker expressions are assigned symbols based on team responses, rather than on speaker expressions.

Sadly, the assignment of symbols to conversations is currently done manually. Under this slow and expensive process, a person watches the video recording of a conversation, types the spoken content and chooses an IDN label for each sentence turn. Because of this arduous process, data generation is very slow and limited in scope. As such, having a system that automatically tags or at least gives fairly good recommendations as to how to tag different parts of conversation would be enormously beneficial in speeding up the tagging process. A current problem in design team research is the lack of data on team interactions. Having a system that helps in generating new data on design teams would enable the development of models to extract patterns of effective teams across different domains and contexts. In recent times, neural networks have made huge strides in the tasks of language translation and text summarization. Neural networks have been able to surpass traditional methods of specific feature extraction by allowing the network to not

only learn to recognize features but also be able to adjust how much weight it gives to each of these features. Recurrent Neural Networks in particular have shown great performance in being able to extract sentence meaning. Because this networks keep in memory information about previous states they can be very effective in understanding sentence context. As expected, neural networks have proven to be effective in dialog act classification and as we explore in this paper also for IDN classification.

2. Previous Work

Almost no work has been done previously on IDN in particular. Nonetheless, for non-context-dependent datasets, past research has been done in statistical methods and some in Gated Recurrent Networks (GRUs) for classification.

2.1. Tagging Using Statistical and Probabilistic Methods

Stolcke at. al proposed a statistical approach for modeling dialogue acts in conversational speech and collected a simplified version of the Switboard Dialogue Act dataset (SWDA).[8] The original SWDA dataset was tagged using the Dialogue Act Markup in Several Layers (DAMSL) tag set, which was designed by the natural language processing community with sponsorship of the Discourse Resource Initiative (Core and Allen 1997).[3] Using the fact that the DAMSL aims to provide a domain-independent framework for dialogue annotation, Stolcke's group developed a simplified tagging scheme of only 42 symbols. While they do admit "that content and task-related distinctions will always play an important role in effective DA labeling," due to the "task-free" nature of the SWDA dataset, the simplification to 42 symbols is effective for conversational speech.
Also, Stockle's team explored several statistical models for automatic tagging. One of those was a bag of words approach. By making use of the idea that some words and phrases serve as indicators of content and structure they were able to find strong correlations between particular phrases and some labels. For instance, they concluded "88.4% of the trigrams '<start> do you' occur in YES-NO-QUESTIONS" (Stockle et. al 2000).
Another probabilistic approach they tried was to predict the dialogue act sequence U^* that had the highest posterior probability given all available information E about a conversation.

$$U^* = \arg\max P(U|E) = \arg\max P(U) * P(E|U)$$

P(U) then can be estimated using a discourse grammar and likelihoods can be empirically estimated from the evidence. Yet another approach is to use a hidden Markov Model (HMM) making certain independence assumptions and assuming that the prior distribution of the dialogue act sequence U is Markovian. Such that

$$P(U_i|U_1, ...U_{i-1}) = P(U_i|U_{i-k}, ...U_{i-1}$$

where k is the order of the Markov process describing U. Under this model, "The HMM states correspond to dialogue acts, observations correspond to utterances, transition probabilities are given by the discourse grammar, and observation probabilities are given by the local likelihoods $P(E_i|U_i)$." A great advantage of using HMMs is the possibility of using fast dynamic programinc techniques to extract model properties. When used along with the Viterbi algorithm for HMMs (Viterbi 1967)[9], they were able to accuratelly calculate the most likely global state that would produce the highest posterior probability.

2.2. Dialogue Act Classification in Domain-Independent Conversations Using a Deep Recurrent Neural Network

A more modern approach using netural networks for deep learning was also tried by Khanpour et. al with relatively high success on the SWDA dataset.[5] Their approach consisted on using a recurrent neural network(RNN) that would generate hidden states for each word in the sentence and then pool these hidden states before feeding into a softmax predictor. They also tested different possible trainings on the word embeddings. They claim to have achieved an accuracy of 71.39% on the SWDA dataset by training Word2vec vectors of size 150 using the Gensim Word2vec package on a large corpus of Google News. Similar accuracies are also achieved by training on a large corpus of Wikipedia articles. Moreover, they also attempted to incorporate dropout in their model and found out that drop out does not have much effect on the final accuracy. Therefore using drop out seems to be unnecessary. Furthermore, they also attempted to use stacking of RNN layers. They claim that the model performs best at 10 layers; however results between using 2 and 15 layers do not appear to vary more than half a percentage making results inconclusive. A possible explanation is that having too few layers does not allow for proper detection of tokens while having too many may produce overfitting.

2.3. SWDA Act Classification using fastText

Previous research at the Stanford Center for Design research showed an improvement in the classification of the SWDA dataset into its 42 primary dialogue acts by using the fastText library developed by Facebook. fastText is a library for efficient learning of word representations and sentence classification. fastText also has a library of 300-length word embeddings trained on a large corpus of Wikipedia files. By using this dataset and doing principal component analysis calculations on it to reduce the vectors to size 150, we were able to achieve an accuracy of 71.4% on the SWDA dataset.

The model trained consisted of a multi-layer unidirectional Long Short Term Memory (LSTM) per sentence. An LSTM is a recurrent neural network (RNN) that can be tuned to remember information dependencies. Figure 1 shows the accuracies and losses obtained in the SWDA dataset. Red represents the training set which is 80% of the SWDA dataset and blue represents the testing set. Each epoch represents an entire pass over the dataset in which the sentences in the training and testing sets are separately shuffled. Batching is done in groups of 50 sentences.

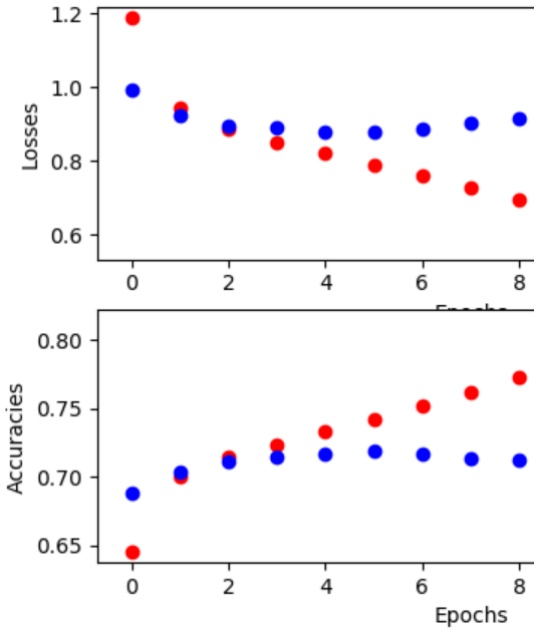

Figure 1. SWDA accuracies and losses per epoch

2.4. IDN classification using a Gated Recurrent Unit

In their CS224s research paper "Dialogue Acts in Design Conversations" (Chan et al.)[1] claim that their model was able to achieve a test accuracy of 64.2% on the IDN dataset. They use GloVe word embeddings passing this representation through a RNN with Gated Recurrent Unit (GRU) cells, and use the final hidden state as their representation for each sentence. Then they use softmax to calculate a probability distribution for the IDN symbols. However, they do not take sentence context into account.

3. Dataset and Features

The main dataset used for this research is the Interaction Dynamics Notation Dataset (IDN dataset) developed by Stanford's Center for Design Research. Unlike the SWDA dataset, the IDN dataset is very much context dependent. For instance, the act of asking a question does not count as an IDN question unless someone else in the group also responds to it. The dataset is still under development but as of this writing, it consists of conversations from 23 different teams for a total of 7230 sentence turns. Each team consists of 2 to 5 people (mostly Stanford students) who are tasked with coming up with a solution to a problem. Problems range in context from finding a way to bring up water from 50 feet underground to improving massive open online (MOOC) education. Currently, the dataset is developed by the arduous and time consuming manual process of paying a human coder to watch each team video, type out what is being said and assign an IDN tag. IDN itself consists of 12 symbols that map the flow of conversation similar to how the notes of a music staff map the flow a song. The 12 symbols are namely: move (M), support(SU), yesand(Y-A), question(Q), humour(H), silence(SI), block(BL), overcoming(OV), deflection(DE), block-support(B-S), yesandquestion(Y-Q), ignored(IG).

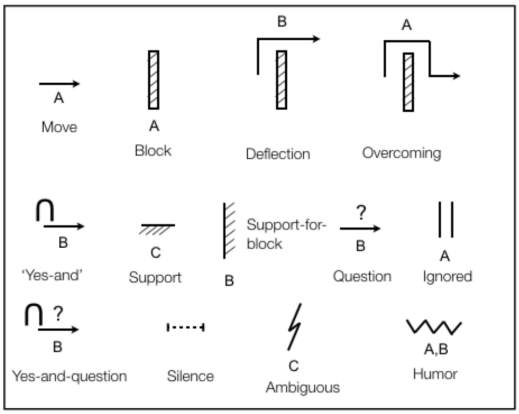

Figure 2. IDN dialogue acts and their symbols

3.1. Data Preprocessing

We follow the standard procedure of using word embeddings to represent each word. Leveraging previous results on the SWDA dataset, we use Facebook's 300-length vectors pre-trained using the fastText approach. As mentioned before, we do principal component analysis on this dataset of 300-length vectors to obtain 150-length vectors for each word. We batch each unit into groups of 50 sentence turns. Then, we randomize the batches to partition the dataset into 80% training, 10% validation and 10% testing using the number 224 as initial seed for consistency across runs. On each epoch we train on the entire training set but shuffle the order of the batches randomly.

3.2. Biases and Noise

A great weakness of the dataset itself is that it has an over-representation of the "move" label which accounts for

about 46% of all sentence turns as can be seen in figure 3. Also since the tagging of the dataset we are training on is done by multiple people there are slight variations in the transcriptions of the speech into text. For instance, humour may be transcribed as an empty sentence or with some contextual evidence in brackets such as "[laughter]" which is left entirely up to the discretion of the human coder.

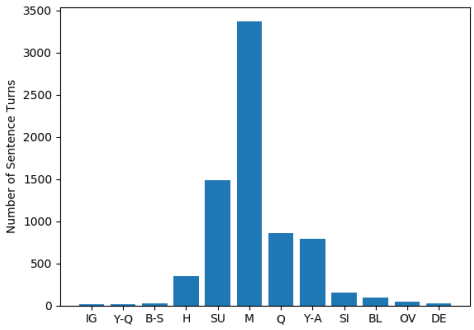

Figure 3. Frequencies of Dialogue Acts

4. Methods

Our overall approach has two parts. First, we train a model on the SWDA dataset and then use transfer learning to obtain scores for each of the the SWDA labels on our IDN dataset. Second, we feed these scores along with the IDN sentence turns into a multi-layer LSTM neural network to predict labels for IDN.

4.1. Training and score-extracting on SWDA

We begin by training a unidirectional 4-layer LSTM model on SWDA. We define the LSTM cell at each time step t to be a set of vectors in $\mathbb{R}^d$ In the equations of figure 4,

$$i_t = \sigma\left(W^{(i)}X_t + U^{(i)}h_{t-1} + b^{(i)}\right)$$
$$f_t = \sigma\left(W^{(f)}X_t + U^{(f)}h_{t-1} + b^{(f)}\right)$$
$$o_t = \sigma\left(W^{(o)}X_t + U^{(o)}h_{t-1} + b^{(o)}\right)$$
$$u_t = \tanh\left(W^{(u)}X_t + U^{(u)}h_{t-1} + b^{(u)}\right)$$
$$c_t = i_t \bigodot u_t + f_{(t)} \bigodot c_{t-1}$$
$$h_t = o_t \bigodot \tanh(c_t)$$

Figure 4. LSTM definition

X_t is the d dimensional vector input at time t, $W^{hh} and W^{hx}$ are weight matrices and σ represents the sigmoid function. Conceptually, i_t is the input gate, f_t is the forget gate, o_t is the output gate, c_t is the memory cell, and h_t is the hidden state.

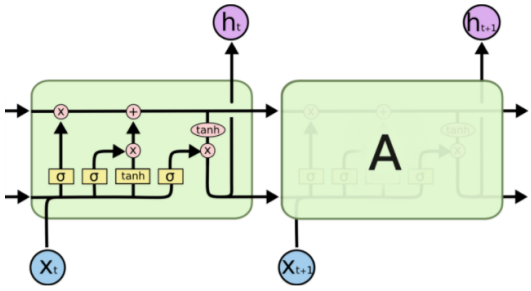

Figure 5. LSTM cell structure and parameters (http://colah.github.io)[2]

We stack multiple LSTM layers by arranging LSTM cells back to back where the hidden layer, h_t of each cell is taken in as input for the subsequent layer in the same time step. The rationale for using multiple layers is that we wish to capture longer dependencies between each sentence word.
Accuracies and losses for this step on the SWDA dataset are shown in figure 1 with red representing the training set (consisting of 80% of SWDA) and blue the testing set.
We then use our SWDA-trained model on the IDN dataset to calculate probability scores for each of the 42 simplified SWDA labels. This step outputs a 42-length vector per sentence turn in the IDN dataset.

4.2. Training on IDN

For the second part, we take as inputs the SWDA scores extracted in the step before along with the IDN sentences to train on the IDN dataset and classify into the 12 different IDN labels.
To represent each word in a sentence, we again make use of 150-length word embeddings pre-trained on Wikipedia using Facebook's fastText. For consistency, we set a maximum of 30 words per sentence and use 0-padding for smaller sentences. We keep track of sentence lengths up to max-length to later on be able to identify final hidden states and bypass the extraneous padding. For our loss operation we use sparse Softmax cross entropy with logits and for our training operation we use the ADAM optimizer.[4] Some of the advantages of using ADAM are that weight updates are invariant to gradient re-scaling and that it naturally performs step size annealing. Under traditional Stochastic Gradient Descent, one would decrease step size over time to prevent overshooting. Thankfully, using ADAM there is no need to set the decrease explicitly also eliminating the need to search for an annealing hyper-parameter.

We take Chan et al.'s implementation of a Gated Recurrent unit as our baseline and implement and compare three more models making use of our previously extracted SWDA scores.

4.2.1 Model 2: Unidirectional Multi-layer LSTM

Drawing inspiration from the model we used to train on SWDA, we also implemented a unidirectional multi-layer LSTM model on IDN in the hopes that it would perform just as well. We perform pooling of the hidden states as shown in figure 6. We use the last-pooling technique in which we gather the hidden states of the last words in the sentence batch after they pass trough the multilayer LSTM. As in SWDA we use 4 layers and we also batch in groups of 50 sentences. Finally to incorporate the SWDA scores we concatenate these scores with our output from the pooling to obtain vector representations of the sentence turns. From this we obtain our logits by passing our vector representations through an affine layer that maps into vectors of size 12 (the number of IDN labels). Shown in figure 7 are the accuracies and losses for this method. Again red represents the training set and blue the test set.

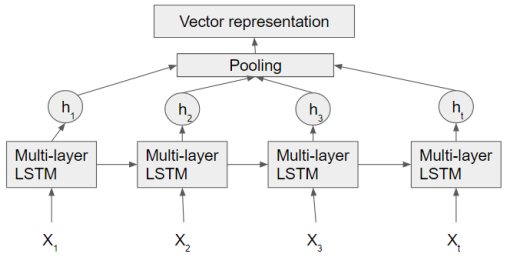

Figure 6. Hidden state Pooling scheme

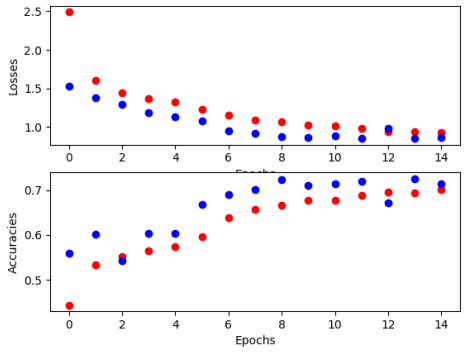

Figure 7. Accuracies and losses for IDN unidirectional method

4.2.2 Model 3: Bidirectional Multi-layer LSTM

This model is very similar to the previous one but instead of passing hidden states from LSTM multilayer stack to the next one unidirectionally, we compute both the forward and backward flow in a bidirectional fashion. This produces two vectors of output hidden states from each stack. One of these vectors is obtained when going in the forward direction and the other when going in the opposite direction. Before pooling we concatenate these vectors. Then we also do last-pooling as before. This way, the pooled hidden state is a concatenation of the last-hidden states from each flow. Accuracies and losses are shown in figure 8. Red represents the training set and blue the test set.

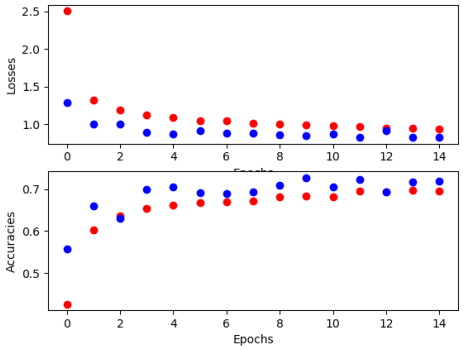

Figure 8. Accuracies and losses for IDN bidirectional method

4.2.3 Model 4: Context-dependent Bidirectional Multi-layer LSTM

For our last model we incorporate features from the sentence before and the sentence after (if existent) within a batch to capture the context of the current sentence. We do this by passing each final hidden vector of the sentences in a batch through an affine layer to compress into 15-length vectors. Then, for each sentence in the batch we look at the 15-length vector of the sentences before and after and concatenate these to the final hidden vector each sentence. The rationale for the compression is so we can treat the 15-length vectors as features that capture general context of a sentence without giving too much weight to the exact order or wording of the sentence.

4.3. Results

4.3.1 IDN Quantitative Results

To report our quantitative results, we evaluated our trained encoder models on our withheld testing set representing 10% of the data. To calculate the loss we used the sparse Softmax cross entropy with logits. Additionally, to initial-

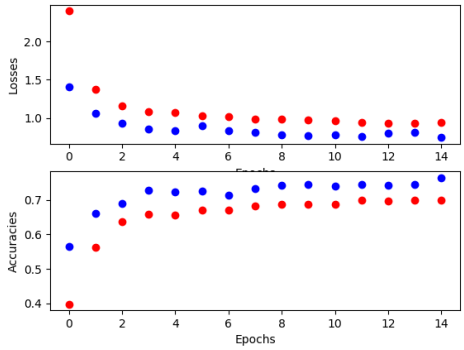

Figure 9. Accuracies and losses for IDN context-dependent method (Red represents the training set and blue the test set)

Model	Test Accuracy	Test Loss	Train Accuracy	Train Loss
Chan et al.'s GRU	64.2%	NA	NA	NA
Unidirectional Multi-layer LSTM	70%	0.65	68%	0.98
Bidirectional Multi-layer LSTM	71%	0.56	68.2%	0.98
Context-dependent Bidirectional Multi-layer LSTM	73%	0.5	68.5%	0.96

Figure 10. Model accuracies and losses

ize the LSTM layer weights, we use the Xavier initializer with random seed of 224.[10] Also for the training operation we use the ADAM Optimizer as explained before.
Figure 10. Shows final results across all our models. From it, we can definitively conclude that our fourth model of context-dependent Bidirectional Multi-layer LSTM performs best.

4.3.2 Discussion of Results

As we can see from the graphs in figures 7, 8, and 9 and the table above, losses decrease consistently over time as we run more epochs. At the same time, there is a trend for accuracies to increase. These patterns demonstrate that the model is effectively learning over time. It is interesting to note how the trends become more stable as we take context into account. For instance, the accuracy lines of our context-dependent model are much smoother and with less sudden jumps than for the simple LSTM model. Moreover, test accuracies tend to be slightly higher than train accuracies across our models. Similarly, the losses for the test set are slightly lower than those for training. These characteristics seem to indicate that our models does not attempt to over-fit on the training data. However, given that we have a small dataset it is hard to conclude how that would translate to a much broader context. By the nature of the design team conversations, there are few restrictions on brainstorming. This means that topics may change suddenly and ideas may not always reach a conclusion. In terms of our model, this means that unforeseen contexts may lead to inaccurate predictions. Furthermore, for our context-dependent model, when we take the average final accuracy after 15 epochs of 200 runs, we get a value of 73.23%. This suggests that the model may have multiple maxima instead of a single absolute maxima to which it can converge. This may explain why there would exist variability across runs even though we set an initial seed of value 224 on each run.

5. Conclusion

We were able to achieve accuracy results on IDN close to those of state-of-the-art for SWDA by pre-training a model on SWDA and then leveraging that output for training a second model on IDN. We also compare different models for training in IDN using our extracted scores from the first SWDA model and conclude that the best current model is a context-dependent Bidirectional Multi-layer LSTM.

6. Future Work

The next future step should be increasing the size of the IDN dataset so we can show how well our results generalize to a broader context. Also with more training data, our model should in theory be able to learn more about real-world conversations and adapt better. Currently, the IDN dataset is still growing, but because of the aforementioned slow and expensive manual process of labeling the data, full collection may take a while to implement. Moreover, there still remains the issue that most dialogue acts within a conversation are "move." Because of its extensive prevalence, the "move" label may skew the model into being more prone to predict "move" too often. An approach we could try in the future to fix this is to make each label have equal amounts of sentence turns when training. However, this pattern is mostly due to the nature of the conversations themselves and not necessarily a systematic bias. Furthermore, the IDN dataset does not explicitly capture levels of energy and positiveness which may be crucial in determining cohesion within a group. Overall, a much richer dataset would certainly help our model learn better.

References

[1] chan et. al. Dialogue acts in design conversations. 2016.

[2] colah. http://colah.github.io/. 2015.

[3] core and james allen. Coding dialogs with the damsl annotation scheme. 1997.

[4] diederik p. kingma and jimmy ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[5] khanpour et.al. Dialogue act classification in domain-independent conversations using a deep recurrent neural network. 2012.

[6] kleinsmann m. valkenburg r. Learning from collaborative new product development projects. 2005.

[7] sonalkar et. al. Developing a visual representation to characterize moment-to-moment concept generation in design teams. 2013.

[8] stolcke et. al. Dialogue act modeling for automatic tagging and recognition of conversational speech. 2000.

[9] viterbi et. al. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. 1967.

[10] xavier glorot and yoshua bengio. Understanding the difficulty of training deep feedforward neural networks. 2010.