

INFORMATION EXTRACTION FROM A MULTI-THREAD BUSINESS CONVERSATION

by

Nikhil Yelamarthi

EE18B142

Under the supervision of

Dr Gaurav Raina and Dr Oshin Anand

Department of Electrical Engineering

Submitted

in fulfillment of the requirements of EE4900: Btech Project

to the



**INDIAN INSTITUTE OF TECHNOLOGY
MADRAS**

JUNE 2021

Abstract

This thesis primarily focuses on developing an end to end module which can process multi-threaded conversations and extract information specific to a domain and intended business task. The task at hand is to assess multiple contexts in the conversation, understand the dialogue flow and utilise that in extracting information from the conversations.

Since the target is free flow dialogue, understanding and maintaining different context flow in a multi-threaded conversation is crucial. The attempt is to build a solution that can connect these contexts and is reflected in the extracted information, taking care of things like negotiations. Three domain- specific entities: price, quantity, and discount are defined, and these are the attributes whose values are required to be extracted from the conversation at the end. If there are more than one product being discussed in the conversation, then our goal is to extract the values of price, quantity and discount for each product.

The solution is designed in the form of three modules. The first module consists of data generation using a state machine approach and data pre-processing. The second module consists of sentence class prediction and entity class prediction, whose outputs are then passed onto the third module. These outputs are then used in the third module to establish context and extract the required information. A method of context index bucketing and product-context vector has been conceptualised to achieve the same.

The overall accuracy depends upon the individual accuracies of the sub-modules and can be improved by increasing the data volume and variety while training and using more advanced classifiers for the individual sub-modules. Also, a few cases are pointed out where the context understanding fails and results in wrong extraction. The overall end to end prediction accuracy is around 80%. While the present solution has been hardwired for a particular case, where only two products are being talked about, it can easily be extended to a multi-product scenario.

Contents

[Abstract](#)

[Contents](#)

[List of Figures](#)

[Abbreviations](#)

1	Introduction	1
1.1	Problem Statement	1
1.2	Solution Approach	2
1.3	Thesis Organisation	3
2	Literature Review	4
2.1	Related works	4
2.1.1	Utterance/Sentence type classification	4
2.1.2	Message disentanglement and Process Mining	5
2.1.3	Entity Classification using word Embeddings	6
2.1.4	Information Extraction with Morphological tools	6
3	Data Generation	7
3.1	State Machine Approach	7
4	Solution Design	9
4.1	Data Preprocessing	10
4.2	Dialogue Classification	10
4.2.1	Sentence Type Class Prediction	10
4.2.2	Entity Class Prediction	14
4.3	Integration Module	15
4.3.1	Context index Bucketing using Sentence Class prediction out-puts	16
4.3.2	Entity Class Correction	18
4.3.3	Product Detection	18

4.3.4	Generation of Product Context Vector	18
4.3.5	Morphological Analysis for Value Extraction	20
Conclusion and Scope for Future Work		22

List of Figures

3.1	A sample conversation generated from the state machine approach . .	8
4.1	A detailed flow diagram of the solution diagram.	9
4.2	Predicted sentence classes for the dialogues in the sample conversation	13
4.3	Model summary of Bi-LSTM Model used for entity class prediction .	14
4.4	Predicted entity classes for the dialogues in the sample conversation .	15
4.5	Generation of context index buckets for the sample conversation . . .	17
4.6	Generation of product index vector for the sample conversation . . .	19
4.7	Values extracted from the sample conversation	21

Abbreviations

NLP	Natural Language Processing
IR	Information Retrieval
IE	Information Extraction
PoS	Part of Speech
LSTM	Long Short Term Memory network
Bi-LSTM	Bi-Directional Long Short Term Memory network
BERT	Bi-Directional Encoder Representations from Transformers

Introduction

1.1 Problem Statement

In a business setting, there are multiple conversational interactions within the company as well as with customers. For example, dialogue between a sales representative and a potential buyer, customer and customer support executive etc. These dialogues have information that can be extracted for further actioning and can also be used to automate tasks and perform analytics for business decisions. Hence the task at hand is to develop a solution which can extract information from a given business conversation. The values of interest in a business conversation are the different products being talked about, their prices, quantity, location of delivery and discount. These are the values which we need to be extracted from the conversation to generate a purchase order.

If the conversation is about one single product, it would be a bit straightforward to detect these entities as there would not be any context change. However if the conversation deals with more than a single product, it becomes tricky. This is because, we need to understand whether a particular dialogue in the conversation is talking about the first product or the second product. The problem statement is

thus to design a solution to extract the values of the entities: price, product, discount and location of delivery from a multi-product, multi-thread business conversation.

1.2 Solution Approach

To solve this issue of understanding the context, the concepts of context index buckets and product context vectors are introduced. These are two different methods of establishing the context in the conversation. These methods are built upon two independent dialogue classification modules, namely sentence type classification and entity type classification. The outputs from these classification modules are used in the above mentioned methods to establish the context in the conversation.

Sentence type classification is a domain-independent classification technique where each sentence is classified into one or more of a few pre-defined sentence types like question, answer, follow-up question etc. while Entity classification is a domain-specific classification method that can be used to predict whether each dialogue is talking about either of: product, quantity or price.

Once the product context vector is established, it could be easily inferred, which product does each dialogue talk about and from the entity classification, we already know which entity the dialogue is talking about. Hence morphological tools are then used to extract the values corresponding to each entity and product in the conversation.

1.3 Thesis Organisation

The rest of the thesis is organised as follows. In Section 2, a detailed literature review is done analysing some of the latest work done in the field and how this thesis builds upon them to solve the desired problem statement. In section 3, the method for data generation for training and testing is explained. In section 4, the complete working of the end-end module is described in detail. Section 4.1 covers the data pre-processing aspects. Section 4.2 explains the dialogue classification modules. Section 4.2.1 explains the Sentence type class prediction while section 4.2.2 explains the Entity type class prediction. The final integration module and value extraction are then finally explained in Section 4.3. The paper ends by concluding the work and mentioning the scope for future work.

Literature Review

2.1 Related works

The solution is based around the concepts of Information Retrerival(IR),Information Extraction(IE), sentence type classification,domain-specific entity classification and process mining. In the rest of this section, related works referring to each individual sub-problem will be discussed.

2.1.1 Utterance/Sentence type classification

Sentence-type classification is an integral part of this solution design as it helps to establish the intent of each dialogue in a conversation. The idea of sentence-type classification/utterance classification is a well- investigated research topic in the field of NLP and IR. Many different classification techniques like statistical approaches[6], hidden Markov models and Support Vector Machines(SVM)[7] have been used for this purpose.

Bhatia et al.[8] proposed a classical machine learning approach for classifying messages and extracted features based on content, sentiment and structural information. However, the features were dependent upon user profile. Qu et al.[2] leveraged this taxonomy proposed in [8] by removing the user-profile dependent features and adding 4 more classes. The authors of [2] have also created a large scale labelled dataset, MSDialogue extracted from a Microsoft Support Forum. This taxonomy developed in [2] was further utilised by Holstrup et al.[1] to perform utterance classification on the dialogues of an online conversation and use the predicted labels to discover process models within the conversation. For mapping the conversations to process models, they used specific pre-defined patterns discovered by looking at a synthetic dataset they have developed. However, they have not mentioned the patterns used and if and how they have dealt with situations where more than one utterance classes have been attributed to a particular sentence.

2.1.2 Message disentanglement and Process Mining

The research done on implementing process mining techniques in analysing conversation is quite limited. According to Holstrup et al.[1] , their's is the first known work to extract process models from unlabelled and unstructured information-seeking data-sets. Since their analysis was based on a multi-party,multi-thread conversation, they have adopted cosine similarity based message stream disentanglement for identifying the separate conversations within a single stream of messages.This kind of a situation commonly occurs on online platforms like Twitch or Glitter. Similarly Tan et al.[5] proposed a method for message entanglement in a multi-party scenario. However, our scenario is a business conversation with a single party talking about multiple products/services, which could be intertwined within every single message. Hence doing message entanglement does not help to entangle the discussion about

each product/service in our case. Hence we have built upon the idea of process mining as introduced in [1] to establish a context-index bucketing and further introduced something called a product context vector for this purpose.

2.1.3 Entity Classification using word Embeddings

For entity detection, we have used a Bi-Directional LSTM model using word embeddings, an idea which was first used for training named entity recognition by Wang et al.[4]. The same idea has been extended to classify entities at a sentence level rather than a word-level classification as done in [4].

2.1.4 Information Extraction with Morphological tools

Information extraction(IE) using morphological tools like Part of speech(POS) taggings and dependency parsing is a common practice in NLP and was first explored in 1990s by many authors like Riloff[9] and Hobbs[11]. In the present day, pre-trained POS tagging and dependency parsing can be exploited using the spacy library in python[10]. This is used in our final integration module to extract the required values once the context is established.

Data Generation

3.1 State Machine Approach

To generate data, a semi-automated state machine approach was followed .A set of states and state paths are defined from which conversations can be generated.Each state represents a particular situation in the conversation and consists of a set of sentences which deliver the same meaning.

Two types of states are defined: Buyer states and Seller states, corresponding to the dialogues of the buyer and the seller respectively. There are 34 buyer states defined and 29 seller states defined. For example, bs01 is a buyer state which represents the intent - ‘Ask for the availability of product A’. This state will thus consist of a set of sentences that will deliver this intent, for example: ‘Do you sell phones?’ ‘I am looking for phones,are they available?’ etc.

An extensive set of state paths is then generated, by connecting these state nodes together in linked lists. The nodes are connected in such a way that, each state path represents a complete meaningful conversation between a buyer and seller. During the generation, one of the state paths is chosen at random, and from each of the

states in that state-path, one particular sentence is again selected at random. What results, is a set of different conversations between the buyer and seller.

The different state paths represent the different kind of ways, a conversation can go. For example, a customer may ask for one product, and after knowing it is not available, he might ask for another product, or the customer may ask for both the products at once, but decide to take only one because the price of the second is high. Similarly, he/she can negotiate for both products together or negotiate for each product separately. Each of these different kinds of conversations is represented by a different state path. Hence by making as many buyer and seller states as possible, it was possible to bring in as much variation in the data as possible.

The below picture shows a snapshot of a sample conversation generated using this method. The same conversation will be referred throughout the document to explain the functioning of the end to end module.

Dialogue	
	Hi is this Kamal sports? I need 100 bats delivered to delhi.
	Yeah, I can supply
	I am also looking for balls.Do you have them?
	Yes,
	How many you require
	We need 200
	Sorry we are low on production, can't take the order
	The cost would be 2000 per bat
	Please give some discount, the price is a bit high
	1750 is the last that can be offered
S	Okay,let's go ahead with the order

FIGURE 3.1: A sample conversation generated from the state machine approach

Solution Design

The following is a detailed flow diagram of the proposed solution. Each module is explained in detail in the subsequent sections.

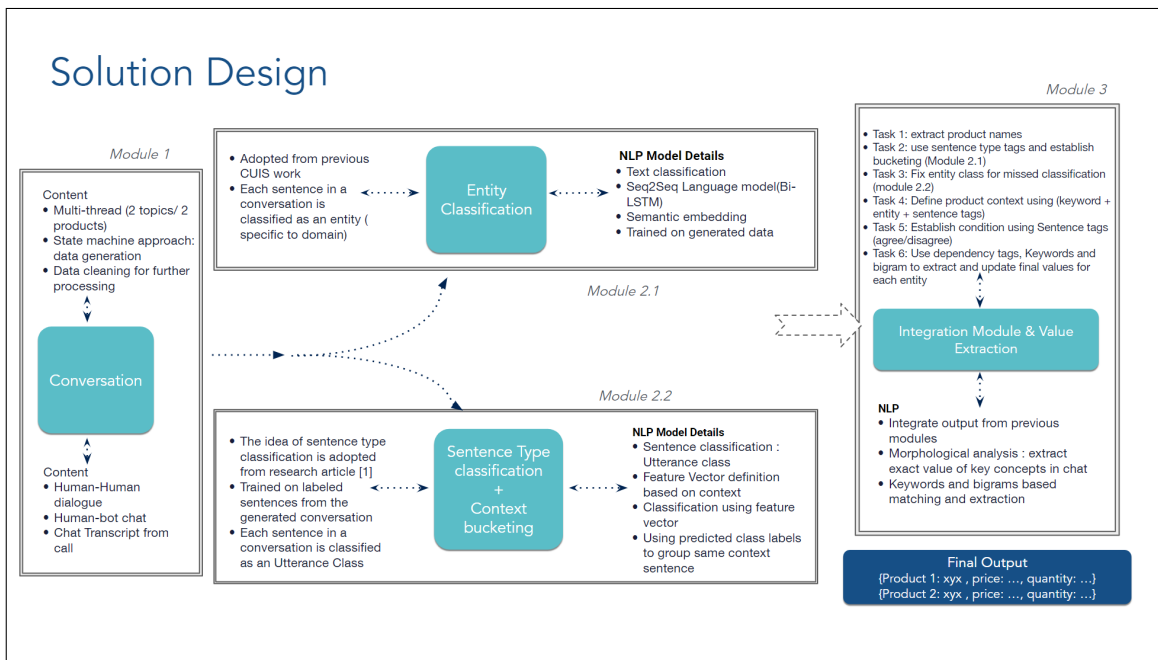


FIGURE 4.1: A detailed flow diagram of the solution diagram.

4.1 Data Preprocessing

The data generated from the data generation module is in plain English language without any grammatical mistakes and spelling errors. Hence only basic data cleaning is done like removing punctuation marks and extra spaces. One particular step done in pre-processing is the conversion of numbers to their corresponding text, using num2words library in python. This has helped in improving training accuracies in the classification modules.

4.2 Dialogue Classification

The second module in the solution model is the dialogue classification. There are two different types of classification modules running in parallel.

4.2.1 Sentence Type Class Prediction

The idea of sentence type class prediction is referred from [2], where the terminology utterance-class is used. They have defined 12 different utterance-classes, and each dialogue in the conversation could be tagged with either of these 12 classes. The same has been adopted in this solution, but with a few modifications. Having realised that some of these classes would not be relevant in a business conversation, we decided to use 8 out of those 12 classes, with a few altered definitions of the classes. The below is the list of classes we used in our module.

Label	Definition	Sentence-Type Class Label
1	OQ	Original Question
2	PA	Potential Answer
3	FQ	Follow-up Question
4	FD	Further Details
5	IR	Information Request
6	PF	Agreement (originally defined as positive feedback in [1])
7	NF	Negation (originally defined as negative feedback in [1])
8	JK	Junk

TABLE 4.1: **Sentence-type class labels**

The sentence-type class labelling has been done semi-automatically as a part of the data generation process. For training the module, first, feature vectors are extracted from the dialogue and then they are passed into a random forest predictor similar to the process followed in [2].

In [2], 15 functions are defined for extracting feature vectors which captured content, structural and sentimental attributes of the dialogue. However, we have figured out that few of those are irrelevant for a business conversation scenario. Therefore it was decided to use 9 out of those 15 functions, few of them slightly redefined to suit the purpose. The following are the functions used for the extraction of the feature vectors.

Feature Name	Description	Type
Initial Utterance similarity	Tf-Idf cosine similarity between a given dialogue and first dialogue in the conversation	Real
Dialogue similarity	Tf-Idf cosine similarity between a given dialogue and the whole conversation	Real
Question	Does the dialogue contain a question mark or any of the following keywords: what, where, when, who, why, how	Binary
Normalised position	The position of dialogue in the conversation divided by the number of dialogues in the conversation	Real
Unique stemmed length	Number of unique words present in a dialogue after removing stop words and stemming all the words	Integer
Buyer statement	Is the dialogue by the buyer	Binary
Thank	Does the dialogue have the keyword thank	Binary
Sentiment scores	Sentiment scores computed using VADER[positive,negative,neutral]	Real
Opinion lexicon	Number of positive and negative words from an opinion lexicon	Numeric

TABLE 4.2: **Functions used for feature extraction for sentence class prediction**

The model was tested on three variations of train and test split. The accuracies in the three cases are as follows:

1: Train test split from similar data variation :

Achieved Accuracy 90%

2. Test data: New sentences, product names, and different semantics

Achieved Accuracy: 87%

3. Test data: Changing the nature of the conversation; using different state paths to generate conversation

Achieved Accuracy: 70%

Increasing data volume and using more advanced classification methods would lead to higher accuracy in each case. Below is a snapshot showing the prediction of the trained model on the sample conversation. All the experimentation is done using the second train-test split described above. This is because the third variation just represents how the model responds to completely different data and the accuracy in such a case can be improved by increasing the data variety.

Dialogue	Utterance Class
Hi is this Kamal sports? I need 100 bats delivered to delhi.	('OQ',)
Yeah, I can supply	('PA', 'PF')
I am also looking for balls.Do you have them?	('OQ',)
Yes,	('PA', 'PF')
How many you require	('IR',)
We need 200	('PA',)
Sorry we are low on production, can't take the order	('NF',)
The cost would be 2000 per bat	('FD',)
Please give some discount, the price is a bit high	('FQ',)
1750 is the last that can be offered	('PA',)
Okay,let's go ahead with the order	('PF',)

FIGURE 4.2: Predicted sentence classes for the dialogues in the sample conversation

4.2.2 Entity Class Prediction

As discussed earlier, three domain specific entities: price, quantity and discount are defined. The entity classes are labelled for the generated data semi-automatically as a part of the data generation process. For training, a bi-LSTM with word embedding is used. The model parameters have been tuned in such a way to get maximum possible validation accuracy using the test data. The following shows the model summary with the final values of the tuned parameters.

Model: "sequential"		
Layer (type)	Output Shape	Param #
embedding (Embedding)	(None, None, 250)	68000
bidirectional (Bidirectional)	(None, None, 128)	161280
bidirectional_1 (Bidirectional)	(None, 64)	41216
dense (Dense)	(None, 3)	195

FIGURE 4.3: Model summary of Bi-LSTM Model used for entity class prediction

The accuracy of the module is around 80% and can be further improved by using more variety of data for training and implementing more advanced classifiers like BERT. The following is a screenshot of the predicted entity class for the dialogues in the sample conversation.

Dialogue	Quantity	Price	Discount
Hi is this Kamal sports? I need 100 bats delivered to delhi.	1	0	0
Yeah, I can supply	0	0	0
I am also looking for balls.Do you have them?	0	0	0
Yes,	0	0	0
How many you require	1	0	0
We need 200	1	0	0
Sorry we are low on production, can't take the order	1	0	0
The cost would be 2000 per bat	0	1	0
Please give some discount, the price is a bit high	0	0	1
1750 is the last that can be offered	0	1	0
Okay,let's go ahead with the order	0	1	0

FIGURE 4.4: Predicted entity classes for the dialogues in the sample conversation

4.3 Integration Module

In the integration module, the predictions from the above two modules and a few morphological tools are used to establish the context and perform value extraction. The integration module consists of four sub-modules which will be explained in the subsequent sections.

4.3.1 Context index Bucketing using Sentence Class prediction outputs

As a first step in the integration module, the conversation is parsed into buckets using the predictions from the sentence-type classification module. The concept of linking sentences based upon their sentence-type is an idea first mentioned in [1]. A few more robust set of rules have been developed for the same, in this thesis, to establish proper context parsing. A dictionary has been defined, which dictates which sentence type can be linked to which sentence type. It can be seen below. This mapping was defined based on intuition, and many changes were done to it, through the course of experimentation on the generated data.

Label	Sentence-Type	Sentence-Types it can link to
1	OQ	PA
2	PA	FQ,PF,NF
3	FQ	PA,FD
4	FD	FQ,PF,NF
5	IR	FD,PA
6	PF	IR,FQ
7	NF	IR,FQ
8	JK	-

TABLE 4.3: **Sentence-type class label linking rules for context bucketing**

While going through the conversation, the module checks whether each sentence could be linked with its previous statement or not. If it can't be linked so, then a

new bucket, is formed, indicated with an increment of the value of the context index. The context index of the first dialogue of the conversation is set to 0 by default. Hence each context index represents a set of dialogues within the conversation that speak about the same context.

When a dialogue has more than one sentence-type tags associated with it, then the dialogue could be placed in more than one bucket, based upon whether or not its different tags link to the previous and next dialogues, respectively. The following is a screenshot showing the context bucketing for the sample conversation.

Dialogue	Utterance Class	context index
Hi is this Kamal sports? I need 100 bats delivered to delhi.	('OQ',)	0
Yeah, I can supply	('PA', 'PF')	0
I am also looking for balls.Do you have them?	('OQ',)	1
Yes,	('PA', 'PF')	(1,2)
How many you require	('IR',)	2
We need 200	('PA',)	2
Sorry we are low on production, can't take the order	('NF',)	2
The cost would be 2000 per bat	('FD',)	3
Please give some discount, the price is a bit high	('FQ',)	3
1750 is the last that can be offered	('PA',)	3
Okay,let's go ahead with the order	('PF',)	3

FIGURE 4.5: Generation of context index buckets for the sample conversation

4.3.2 Entity Class Correction

Sometimes, the entity classes prediction might fail for a particular dialogue, and it may not be detected either as a price, quantity or discount related statement. In that case, the entity class of that dialogue is made equal to the entity class of previous dialogue in the context-index since the notion of the buckets is that all dialogues in the bucket talk about one particular context. If a dialogue is the first dialogue of the bucket, and its entity class hasn't been predicted, then a simple keyword search is used to try establish the entity. In this way, the end to end prediction accuracy is increased, even if the entity cannot be detected by the entity class prediction module.

4.3.3 Product Detection

In this step, the code parses through the conversation and finds out the list of products, the conversation is talking about. It is assumed that whenever the buyer asks about a new product, one of the associated utterance classes tag would be OQ, Original Question. This is because the buyer is asking for something, which he has not mentioned till then. Hence the dialogues labelled as OQ are analysed using the dependency tags and part-of-speech tags from spacy to extract the names of all the products being discussed in the conversation. For instance, in the running example of the sample conversation in this paper, the product list generated would be [bat, ball]

4.3.4 Generation of Product Context Vector

Now since it is known, what are the products being discussed in the conversation, the next task is to associate each of the dialogues in the conversation to one or multiple

products. To achieve this, something called as the product context vector is defined. Keyword matching is used to keep track of where the product names are being referred in the conversation. The Affirmation(PF) and Negation(NF) tags are then used to understand if a particular product is being dropped from the conversation at any point or not. If a particular dialogue has an 'NF' tag in it and more than one product is mentioned in that dialogue, then a nearest vicinity keyword search is used to understand, the 'NF' corresponds to which product. For this purpose a set of negation keywords are predefined: 'not', 'don't', 'can't', 'cannot', 'aren't', 'out'. For example, in the following dialogue: 'Apples are not available, but oranges cost 20 each.', the predicted sentence classes are 'NF,FD'. The code in this case sets the product index from the next sentence as orange, thus understanding that the 'NF' corresponds to apples.

The following is a snapshot of the detected product context vector for the running example of the sample conversation.

Dialogue	Utterance Class	context_index	price	quantity	discount	product_index
Hi is this Kamal sports? I need 100 bats delivered to delhi.	('OQ',)	0	0	1	0	['bat']
Yeah, I can supply	('PA', 'PF')	0	0	0	0	['bat']
I am also looking for balls. Do you have them?	('OQ',)	1	0	0	0	['ball']
Yes,	('PA', 'PF')	1,2	0	0	0	['ball']
How many you require	('IR',)	2	0	1	0	['ball']
We need 200	('PA',)	2	0	1	0	['ball']
Sorry we are low on production, can't take the order	('NF',)	2	0	1	0	['ball']
The cost would be 2000 per bat	('FD',)	3	1	0	0	['bat']
Please give some discount, the price is a bit high	('FQ',)	3	0	0	1	['bat']
1750 is the last that can be offered	('PA',)	3	1	0	0	['bat']
Okay, let's go ahead with the order	('PF',)	3	1	0	0	['bat']

FIGURE 4.6: Generation of product index vector for the sample conversation

4.3.5 Morphological Analysis for Value Extraction

At this stage, the program knows, which entity (price/quantity/discount), each dialogue is talking about and also which particular products each dialogue talks about. So, in this step, morphological tools like dependency tagging and POS tagging from spacy are used along with bi-gram analysis to extract the values from each dialogue and associate it with its corresponding product and entity.

In case there is more than one entity tag detected for a particular dialogue, nearest vicinity keyword search is performed to associate the values extracted from the sentence to its particular entity. For this purpose, a set of keywords are pre-defined for each of price, quantity and discount:

Price-Keywords: rupee, dollar, \$, rs, cost, price.

Discount-Keywords: discount, %, percent, percentage.

Quantity-Keyword: piece, unit, kg, box, carton.

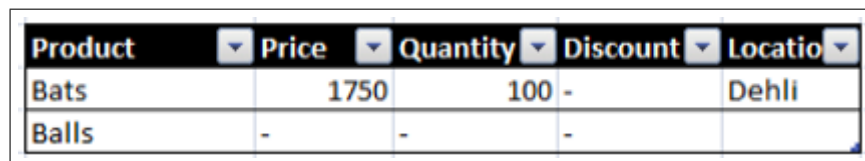
For example, in the following sentence: ‘If you take 100 more pieces, the price will come down to Rs 500 each’, the predicted entity tags are both quantity and price. In this case, the code is able to link 100 as the value of the quantity and 500 as the quantity of the price. Also, if a value follows or is followed by the following Increment-Keywords: [more, additional, extra], then the value of that entity is added to the last-mentioned value of that entity. For example, in the sentence: ‘I will take 50 more headphones’, the program updates the last detected quantity of headphones by adding 50 to it.

Hence in this manner, the values of the extracted entities are updated to a list. Such lists are maintained for each entity for each product. The extraction of the delivery location is done by simply doing a keyword search through the entire conversation,

against an existing list of 1500 Indian city names.

Whenever the code comes across a dialogue with a negation(NF) tag associated with it, the last value of the corresponding entity and product is popped out from the list. Hence the last value present in the list is the final value extracted for each product and entity. In case there is a negation (NF) tag with the length of the product index greater than one, then again nearest vicinity keyword search is used to associate the negation with the entity of a particular product.

The following is a screenshot of the extracted values from the running example of the sample conversation. Here the final price of the bats after the negotiation is rightly found out and also the price and quantity of the balls is not shown as the seller was out of stock for balls.



Product	Price	Quantity	Discount	Location
Bats	1750	100	-	Dehli
Balls	-	-	-	

FIGURE 4.7: Values extracted from the sample conversation

Conclusion and Scope For Future Work

The solution developed seems to work well on the generated test data and provides a prediction accuracy of around 80%, which can be further increased by using a wider variety of data and more advanced classifiers for the prediction module. We have thus been successful in generating a method to establish the context in a multi-product, multi-thread conversation.

Though the conversation is unlabelled and unstructured, and has information about each product intertwined in it's messages, we are able to extract the entities corresponding to each of the products. The code is also able to handle negotiations and extract the correct value of each entity of the product after all the negotiations in most of the cases. The solution could also be used in other information seeking multi-thread conversations, by defining new entities according to the type of the conversation, as the entity prediction module is the only domain-specific module in the whole solution.

The following are a few pointers for further improvement and study:

- Since the data generated was clean English data, we did not have to handle any grammatical and spelling errors. These aspects can be added to the pre-processing module to handle raw data from an actual conversation. This becomes essential as many of the morphological tools used rely on grammatical correctness.
- The present method of using morphological tools for extracting the product names[Section 4.3.3] has led to a few fault cases, and thus a more robust language model could be used for the purpose (Present accuracy - 90%)

- When the product name is a compound noun (like paper plate or ruled book), A pre-processing module must be added for the integration module to work as expected.
- The present code developed is hardwired to handle a 2 product conversation. The same concept can be extended to a multi-product conversation by making a few changes to the functions used in the integration module[Section 4.3]. The first two modules would not have to be changed.

It is hoped that this work will provide the required impetus for future studies on the subject.

References

- [1] Alexander Bøgely Holstrup, Lasse Starklit, and Andrea Burattin. Analysis of information-seeking conversations with process mining. In *Proceedings of the 2020 International Joint Conference on Neural Networks*, United States, 2020. IEEE. 2020 International Joint Conference on Neural Networks, IJCNN 2020 ; Conference date: 19-07-2020 Through 24-07-2020.
- [2] Chen Qu, Liu Yang, W. Bruce Croft, Yongfeng Zhang, Johanne R. Trippas, and Minghui Qiu. User intent prediction in information-seeking conversations. In *Proceedings of the 2019 Conference on Human Information Interaction and Retrieval*, CHIIR '19, page 25–33, New York, NY, USA, 2019. Association for Computing Machinery.
- [3] Paige Adams and Craig Martel. *Conversational Thread Extraction and Topic Detection in Text-Based Chat*, chapter 6, pages 87–113. John Wiley Sons, Ltd, 2010.
- [4] P. Wang, Yao Qian, F. Soong, Lei He, and Zhao Hai. A unified tagging solution: Bidirectional lstm recurrent neural network with word embedding. *ArXiv*, abs/1511.00215, 2015.
- [5] Ming Tan, Dakuo Wang, Yupeng Gao, Haoyu Wang, Saloni Potdar, Xiaoxiao Guo, Shiyu Chang, and Mo Yu. Context-aware conversation thread detection

- in multi-party chat. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6456–6461, Hong Kong, China, November 2019. Association for Computational Linguistics.
- [6] Andreas Stolcke, Noah Coccaro, Rebecca Bates, Paul Taylor, Carol Van Ess-Dykema, Klaus Ries, Elizabeth Shriberg, Daniel Jurafsky, Rachel Martin, and Marie Meteer. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Comput. Linguist.*, 26(3):339–373, September 2000. ISSN 0891-2017.
- [7] D. Surendran and Gina-Anne Levow. Dialog act tagging with support vector machines and hidden markov models. In *INTERSPEECH*, 2006.
- [8] Sumit Bhatia , Prakhar Biyani , and Prasenjit Mitra . Classifying user messages for managing web forum data, 2012.
- [9] Ellen Riloff. Automatically generating extraction patterns from untagged text. 10 1999.
- [10] Facts figures. retrieved 26march 2020 [online]. URL <https://spacy.io/usage/linguistic-feature>.
- [11] Jerry R Hobbs. The generic information extraction system. In *Fifth Message Understanding Conference (MUC-5): Proceedings of a Conference Held in Baltimore, Maryland, August 25-27, 1993*, 1993.