Resolving Abstract Anaphora Implicitly in Conversational Assistants using a Hierarchically-stacked RNN

Prerna Khurana, Puneet Agarwal, Gautam Shroff, Lovekesh Vig
TCS Research
Noida, Uttar Pradesh, India
prerna.khurana2,puneet.a,gautam.shroff,lovekesh.vig@tcs.com

ABSTRACT

Recent proliferation of conversational systems has resulted in an increased demand for more natural dialogue systems, capable of more sophisticated interactions than merely providing factual answers. This is evident from usage pattern of a conversational system deployed within our organization. Users not only expect it to perform co-reference resolution of anaphora, but also of the antecedent or posterior facts presented by users with respect to their query. Presence of such facts in a conversation sometimes modifies the answer of main query, e.g., answer to '*how many sick leave do I get?*' would be different when a fact '*I am on contract*' is also present. Sometimes there is a need to collectively resolve three or four such facts. In this paper, we propose a novel solution which uses hierarchical neural network, comprising of BiLSTM layer and a maxpool layer that is hierarchically stacked to first obtain a representation of each user utterance and then to obtain a representation for sequence of utterances. This representation is used to identify users' intention. We also improvise this model by using skip connections in the second network to allow better gradient flow. Our model, not only a) resolves the antecedent and posterior facts, but also b) performs better even on self-contained queries. It is also c) faster to train, making it the most promising approach for use in our environment where frequent training and tuning is needed. It slightly outperforms the benchmark on a publicly available dataset, and e) performs better than obvious baselines approaches on our datasets.

KEYWORDS

Abstract Anaphora resolution; Hierarchical RNNs; Conversational Assistants

ACM Reference Format:
Prerna Khurana, Puneet Agarwal, Gautam Shroff, Lovekesh Vig. 2018. Resolving Abstract Anaphora Implicitly in Conversational Assistants using a Hierarchically-stacked RNN. In *KDD '18: The 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, August 19–23, 2018, London, United Kingdom.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3219819.3219915

1 INTRODUCTION

Support organizations such as Human Resource Department in large enterprises, Customer care division of Insurance companies, etc. that serve large number of users are often flooded with queries. It becomes harder to manage such a business function when the users are geographically distributed, since there is a need for consistency and correctness of responses to the users' queries, against a given set of policy documents as background knowledge base. A modern approach to run such a business function is via a conversational assistant, which can answer most of the questions, and would bring human support staff into a customer's conversation only if needed due to various reasons, such as model's entropy is high or user is not happy etc.

Tata Consultancy Services (TCS), which has about 390,000 employees distributed across various geographical locations, has deployed a conversational assistant for all its employees to ask HR policy related questions [11, 27][1]. Every query asked to such a system needs to be completely self contained, and cannot automatically resolve references with respect to previous question asked by the user to the same system. However, we observed that in practice when interacting with such a conversational assistant, users often refer to their antecedent or posterior utterances.

Based on the analysis of users' conversation with the conversational system, one can say that system needs to resolve three different types of utterance references, shown as x_1, x_2 and x_3 in Table 1. In the first type of scenario, an entity is mentioned in first utterance, for example 'flexi leave' is mentioned in u_1^1 of Table 1, which is referred to via a pronoun in next utterance, for example 'it' in u_2^1. The second type of scenario occurs in situations when the system is expected to take into account multiple different previous user utterances before answering next query, as shown in x_2 of Table 1. Here, user first asserts three facts one after the other u_1^2, u_2^2, and u_3^2, which lead to a complex situation, and then she asks a question '*What should I do?*' which has no meaning without its antecedent utterances. Here, it is very important to note

[1]Approximately 30k thousand queries have been asked by about almost 6,000 people in last few months, on this system. Approximately 1.2% of the responses were disliked by users.

Session and Utterance IDs	**User Utterance**	**Utterance / Reference Type**	**Expected System Response**
x_1 and u_1^1	What is a flexi leave?	Entity - *flexi leave*	**Ans:** *Flexi Holidays are paid ...*
x_1 and u_2^1	What is the eligibility criterion for it?	Pronoun Reference	**Ans:** *Flexi Holiday is applicable to all full time ...*
x_2 and u_1^2	I am new joinee	Ant. Fact - 1	**Fact Response:** *OK*
x_2 and u_2^2	I am not well and my doctor advised 15 days of rest	Ant. Fact - 2	**Fact Response:** *Tell me more ...*
x_2 and u_3^2	I don't have sufficient leave balance	Ant. Fact - 3	**Fact Response:** *Hmm... I see*
x_2 and u_4^2	What should I do?	Query	Ans: *Employees with less than 2 years of service may be granted ...*
x_3 and u_1^3	I have recently adopted a baby	Ant. Fact - 1	**Fact Response:** *Hmm... I see*
x_3 and u_2^3	Can I apply for adoption leave?	Query	**Ans:** *All full time employees can ...*
x_3 and u_3^3	Currently I am on leave without pay for dependant care	Post. Fact - 2	**Ans:** *While on Leave Without Pay you cannot apply ...*

Table 1: Examples of Abstract Anaphoric Intent Intent Identification

that the answer to the same query would be different if the first fact u_1^2 '*I am new joinee*' was not present in the session of conversation. Third example of such scenario requires the system to resolve a posterior fact to an antecedent query, as shown in session x_3 of Table 1. Here, the user asks a query u_2^3 after asserting the fact u_1^3, but modifies the situation in next utterance u_3^3 with another fact leading the system to modify the answer which was given at the end of u_2^3.

To the best of our knowledge there is no prior work in research literature which attempts to resolve all these types of utterance references together, especially when utterance sequence length is more than 2. In this paper, we attempt this problem with an utterance sequence length of upto 5 utterances, and also release a sample dataset for further exploration by the research community.

It is hard to solve this problem since, most of the known methods propose to apply standard NLP algorithms to identify the entities and then associate it to subsequent pronoun occurrence, which works for pronoun resolution. However, for fact resolution such approaches cease to perform well. To solve this problem we propose to map every sequence of user utterances to a canned answer from a prior list of answers. We refer to this problem as '*Abstract Anaphoric Intent Identification*' problem, and present a novel solution to this problem, by using a sequence of sequence model. Here, first its a sequence of words which forms an utterance and then a sequence of utterances which forms a session of users' conversation, we jointly learn the representation at these two levels by using a variant of Hierarchical Attention Network(HAN) [40]. We also introduce skip connection in the second network and demonstrate that our model not only performs well on our datasets but also on a publicly available dataset. In addition to this, we also present benchmarks on our datasets using standard baseline approaches as applicable to this problem.

The *principal contributions* of this paper are as follows:

I. **Abstract Anaphoric Intent Identification** We describe a practical problem in the context of conversational systems, referred to as *Abstract Anaphoric Intent Identification*, based on the learnings and usage pattern of a conversational assistant deployed for thousands of users, in TCS (Employer of authors).
II. **Deep Hierachical Maxpool Model** (DHMN) We present a hierarchical neural network model that captures both word level as well as sentence level patterns, and also captures contextual dependencies within a sequence of user utterances, in its representation of users' session of conversation. Therefore it becomes possible to resolve the abstract anaphora beyond a pair of user utterances. This model is versatile enough to resolve all types of references as shown in Table 1. This is accomplished by using a combination of a bi-directional LSTM with a maxpool layer on top, and such a framework is organised in a hierarchical manner. We also introduce skip connections in this network, which results in better performance.
III. **Evaluation** We demonstrate results on many different baseline algorithms and demonstrate the benefit of using a hierarchical structure over a flat one. We also show the benefits of using a character to word network that improves performance of the model, when encountered with misspelled words.

We also found that in addition to solving the abstract anaphoric intent identification problem, our proposed model DHMN performs better than the earlier model [11, 27] on the self contained queries also, and takes lesser time to retrain. We therefore intend to deploy the new model proposed in this paper in the production environment. Also, time to re-train the model becomes very important because we frequently train many such models on the Cloud deployment of our solution as described in [27]. Also, in order to handle the misspelled words, we build a separately learned character to word network that can predict the closest word from the word vocabulary, which is another and most frequently observed issue in the production environment of our solution.

In the subsequent sections of this paper, we first present a formal description of the problem in Section 2. We then present a perspective emerging from a survey of recent research publications related to the problem of Abstract Anaphoric Intent Identification in Section 3. Thereafter, we present our proposed approach to solve this problem in Section 4, which uses a Deep Hierarchical Maxpool Network. After that in Section 5 we first describe various datasets used for experimental analysis in this paper, followed by description of our approach on how to create seed data for such a problem. Finally, in Section 6 we present the results of our analysis, which has motivated us to deploy this approach in our production environment.

2 PROBLEM DESCRIPTION

Problem Description: An input (x_i) to *Abstract Anaphoric Intent Identification* problem, comprises of a sequence of user utterances $x_i = \{u_1^i, ..., u_l^i\}$, taken from a session of conversation of user with a conversational system. Here, $0 < l \leq l_{max}$, an utterance can be an assertion of a fact by the user, e.g., $u_1^2 =$ "*I have recently adopted a baby*" taken from Table 1 or a query, e.g., $u_2^3 =$ "*Can I apply for adoption leave*", or a combination of these, e.g., "I have recently adopted a baby, Can I apply for adoption leave. Currently I am on leave without pay for dependant care.", which is completely self contained. Note: its not necessary for an input x_i to have a query type utterance. We assume that every such sequence of user utterances (x_i) has a corresponding target class, referred to as user's intent y_j, with $y_j \in Y$. Here, Y is a set of all intents, and $|Y|$ is observed to be between 100 and 500. For every such intent y_j, there is an answer configured in the system a_j, which is shown to the user as a response to x_i. Therefore, for a given input x_i, the model at inference time should be able to identify user's intent y_j, and show corresponding answer a_j.

Training Data Description: The training data D_{trn} consists of many training tuples d_i, comprising of input (x_i) and its corresponding intent y_j, i.e., $d_i = \{x_i, y_j\}$. The training data contains many semantically similar sequences of user utterances with the same intent, i.e., many different x_i map to the same intent y_j. Some of the training tuples contain only one utterance in its x_i, and corresponding y_j. The training data contains at-least n_{min} such training tuples for every intent in the dataset, i.e. $\forall y_j \in Y$. Other types of training tuples contain more than one user utterance, and number of all such possible sequences of user utterances can be very large, and therefore training data D_{trn} can contain only a representative set of such sequences.

We model this problem as a multi class classification problem, in which for any input x_i we predict the class for a sequence of utterances and retrieve the answer a_j accordingly by maximizing $y_j = \underset{y_i \in Y}{\operatorname{argmax}} P(y_i|x_i)$.

System Description: Users engage with the system by giving natural language utterances one after the other. The *Conversational FAQ Bot* needs to give a response to the user after every user utterance, and it should take into account a maximum of l_{max} last utterances (from start of current *session of conversation*) for deciding the next response from the system. For example, first a user passes u_1^3 as input. Therefore sequence of user utterance x_3^1 at time t_1 would be $\{u_1^3\}$. Next, the user passes u_2^3 as input, and sequence of user utterance x_3^2 at time t_2 would be $\{u_1^3, u_2^3\}$, and likewise at time t_3, $x_3^3 = \{u_1^3, u_2^3, u_3^3\}$. System should respond in natural language after every input from the user, i.e., after a fact type utterance from the user, suitable response should be shown. Note: If there is a gap of t_{min} time between two consecutive utterances, we assume the second utterance as start of a *session of conversation*.

3 RELATED WORK

Abstract anaphoric resolution requires establishing a relationship between an antecedent and its anaphor. Generally, most tasks involve resolving a noun phrase to an entity or an object. This type of anaphoric resolution is commonly referred to as *coreference resolution* and has been typically solved using a mention-ranking model as proposed by authors of [5, 6, 34]. Similarly, if the reference is made to a fact or an event, it is referred to as *abstract anaphoric resolution* as done in Ana et al. [18]. They learn the representations of an abstract anaphor and its antecedent in a shared space using a Siamese neural network. These representations are then used to learn a ranking model that scores the candidate antecedents for a given anaphor. However, the aforementioned line of work is different from what we are trying to attempt, we do not learn a model that ranks an antecedent and its anaphor in a chat based user interface. We also do not construct our dataset which contains annotated anaphor and antecedent pairs. Mainly because the reference resolution in these approaches is limited to only two user utterances, while we need to resolve across many different utterances. Also these approaches don't attempt to resolve posterior references, as described via example x_2 of Table 1. Another key difference with the prior papers along this line of work is that our training data contains many semantically similar sequences of user utterances which lead to the same answer, and we are required to choose from these predefined answers, whereas prior approaches rank the answers based on similarity between the question and the answers.

In our proposed approach, we select a response from a repository of responses. These are also of two types :- single turn retrieval and multi turn retrieval. Previous versions of bots [10, 11, 35] have employed methods like matching user input with appropriate responses, or classification of a user query to fetch an appropriate answer. These methods do not consider previous user inputs, however in this paper we deal with a multi turn retrieval bot, i.e., responses are based on multiple antecedent or posterior utterances. Multi-turn retrieval bots have been discussed in [1, 36, 44]. Zhou et al. in [44] generate embeddings of utterances and of response, and select the appropriate response based on a similarity metric. In [36] the response is matched to each utterance in a conversation, to find the suitable response to a sequence of utterances. We however, do not use an approach that relies on text matching. Our data is available in such a form that we have several sequences of user utterance belonging to one class, and hence is more suitable for a classification kind of task instead of matching utterances to a potential response. Hence datasets like the Ubuntu Corpus[17] is not suitable for our task. Also, as shown in our prior work[11], the question and answer similarity based approach does not improve the classification results for our kind of datasets. Further, all these approaches are retrieval based while we employ a classification based approach as described in the next paragraph.

Dialogue and Question Answering: Broadly there are six types of conversational systems present in research literature. a) First type of systems use a classification approach [11], which is similar to ours. Next, b) another line of papers use retrieval based approaches, involving use of information retrieval techniques. Various approaches described in the previous paragraph [10, 17, 26] are of this kind. In the third approach c) researchers have used generative models to answer users' questions. In this line of work, sequence to sequence models have been used by researchers [31] to generate answers against user queries. Such approaches are also not applicable in our setting, since we need to show pre-approved answers only. The fourth type of approach d) employs deep reinforcement learning[15]. Next, and very popular line of work involves e) usage of knowledge graphs to answer various questions from users [30]. These are mostly limited to factoid retrieval from knowledge graphs. Here, both open domain [30] and close domain [27] question answering are prevalent.

Finally, the sixth line of work f) uses automatic paragraph comprehension [39] to answer users' questions, approaches in this line of work use a memory network as done by Sukhbaatar et al.[29]. This has two main issues in our requirement of answering FAQ on Human Resource policies of an organization: 1) that the policy documents are too complex for this kind of approach and 2) the answers to be shown to the users cannot be formed via sentence selection type approaches, but need be reviewed by legal teams a priori.

Classification & Hierarchical Nwks.: Deep learning has achieved impressive results in text classification [42], both using recurrent and more recently convolutional architectures that employ word level modeling. Zhang et al. [37] apply character-level CNNs for text classification and demonstrated better classification performance due to superior resolution of words with multiple colloquial spellings. Similarly [32] proposed tree-structured LSTMs to exploit sentence structure in order to obtain better semantic representations. Several recent studies have even explored the combination of LSTMs and CNNs for text classification in order to exploit the advantages of both architectures [41] [4] [23].

Both attention and maxpool layers have been used interchangeably in hierarchical networks[38, 40]. The idea behind this was to pick up words or sentences that played a key role in classification of the document. Similarly, Lai et al.[13] show that employing a maxpooling operation over the hidden states obtained from LSTM enabled the capture of pertinent information across the entire text. Effectively, the maxpooling operation acts as an attention mechanism on the most important latent semantic factors. The time complexity of the pooling layer is O(n).

For document classification, LSTM or CNN models that do not exploit hierarchical structure, tend to perform poorly [40]. Long texts or paragraphs are a challenge for LSTMs, and to adequately model complex long documents, it often helps to learn the document representation hierarchically. This can be achieved by capturing the document structure at both the word and sentence level, thereby capturing both syntactic and semantic information [16]. Other approaches which exploit the same hierarchical structure of the documents are described in [14, 21, 33]. Zhao et al. [43] als utilize this approach to first encode word phrases and then to encode sentences. Our approach is along similar lines whereby we first represent user utterances using an embedding, and then form a representation of a sequence of utterances. To the best of our knowledge, this is the first attempt at modeling a conversation using a hierarchical network, to resolve abstract anaphoric references implicitly.

4 DEEP HIERARCHICAL MAXPOOL NETWORK

In this section, we describe the neural network model architecture that we use for the problem described in Section 2. We use a deep neural network comprising of three networks which are hierarchically connected to produce a representation/embedding of the sequence of user utterances $e(x)$. The first network, referred to as *Character to Word Encoder Network*, is used for learning a representation of words $e(w)$, which takes sequence of characters $w = \{c_1, ..., c_n\}$ as input. The word representations taken from this network is then passed to next network which learns to represent user utterance, i.e, $e(u)$, this network is referred to as *Word to Utterance Encoder Network*. Here, utterance is a sequence of words, $u = \{w_1, ...\}$. Finally the third network, referred to as *Utterance to Session encoder Network*, takes the utterance representation and learns to represent the sequence of user utterances, i.e., $e(x)$. This is then input into a final softmax layer for classification or intent identification.

We call this network as *Deep Hierachical Maxpool Network* (DHMN), since a maxpool layer is used in all the three networks, as shown in Figure 1. Details of these networks and learning procedure are given in the subsequent paragraphs of this section. This neural network architecture is inspired from the approach presented by Yang et al. in [40].

4.1 Word to Utterance Encoder Network

Embedding Layer: A user utterance u_i is a sequence of words of varying lengths, $u_i = (w_1, w_2, ...w_n)$, and the maximum length of a user utterance is n_{max}. We represent every user utterance as a sequence of one-hot encoding of its

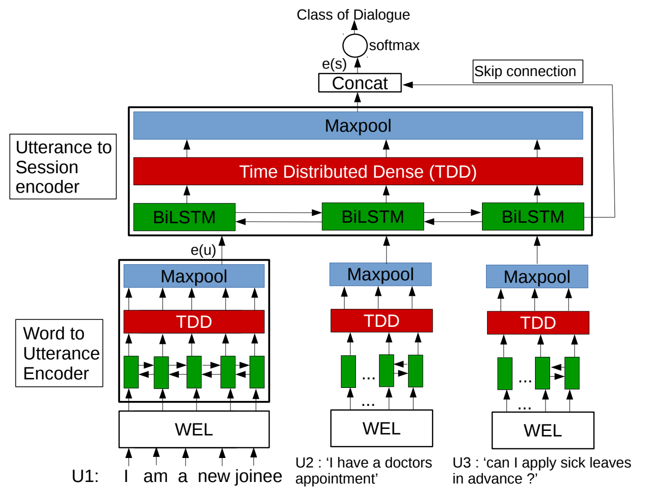
Class of Dialogue
softmax
e(s)
Concat
Skip connection
Utterance to Session encoder
Maxpool
Time Distributed Dense (TDD)
BiLSTM
e(u)
Word to Utterance Encoder
TDD
WEL
U1: I am a new joinee
U2 : 'I have a doctors appointment'
U3 : 'can I apply sick leaves in advance ?'

Figure 1: Deep Hierachical Maxpool Network

words, of length n_{max} with zero padding. Such a sequence becomes the input to the embedding layer which we refer to as WEL in figure 1. The output of this layer is a sequence of word embeddings of the words of a user utterance , i.e., $u_i = \{e(w_1), e(w_2), ..., e(w_{n_{max}})\}$. We first learn word embeddings $e(w)$ using word2vec model [19] on all the queries, answers, and the related policy documents together. These word embeddings become the initial weight of the Embedding layer, which is the first layer in *Word to Utterance Encoder Network*. Weights of this layer (i.e., w2v) also get updated via back-propagation during training of rest of the model.

BiLSTM Layer: The next layer in this network comprises of Bidirectional LSTM Units, as shown in Figure 1 by green units. As a background the LSTMs or Long Short Term Memory networks [8] are a variant of RNNs (Recurrent Neural Networks). LSTMs are designed to mitigate the issue of vanishing gradient, which occurs when RNNs learn sequences with long term patterns. A user utterance returned by *Embedding layer*, is represented as a sequence of vectors at each time-stamp, i.e., $u_i = \{e(w_1), e(w_2), ..., e(w_{n_{max}})\}$, which is input to the BiLSTM layer. This sequence is given as input in both forward and reverse order, as a result at every word in the query it retains the context of other words both on left and right hand side. The output of LSTM unit is controlled by a set of gates in $\mathbb{R}^d$ as a function of the previous hidden state h_{t-1} and the input at the current time step v_t as defined below:

$$\begin{aligned}
\text{input gate}, i_t &= \sigma(\theta_{vi} v_t + \theta_{hi} h_{t-1} + b_i) \\
\text{forget gate}, f_t &= \sigma(\theta_{vf} v_t + \theta_{hf} h_{t-1} + b_f) \\
\text{output gate}, o_t &= \sigma(\theta_{vo} v_t + \theta_{ho} h_{t-1} + b_o) \\
\text{candidate hidden st.}, g_t &= tanh(\theta_{vg} v_t + \theta_{hg} h_{t-1} + b_q) \\
\text{internal memory}, c_t &= f_t \oplus c_{t-1} + i_t \oplus g_t \\
\text{hidden state}, h_t &= o_t \oplus tanh(c_t)
\end{aligned} \tag{1}$$

Here, σ is the logistic sigmoid function, $tanh$ denotes the hyperbolic tangent function, and $\oplus$ denotes the element wise multiplication. We can view f_t as the function to decide how much information from the old memory cell is going to be forgotten, i_t to control how much new information is going to be stored in the current memory cell, and o_t controls output based on the memory cell c_t.

Time Distributed Dense layer: Output of the BiLSTM layer is T hidden states, one at every timestep, which is then passed through a Time Distributed Dense (TDD) layer which is same as applying a fully connected dense layer on the output of BiLSTM at each timestep separately and get output at every timestep. The weights of this TDD layer is same across all timesteps.

Maxpool layer: Output of the TDD layer is then passed through a maxpool layer that acts as a form of attention layer of the network and picks up the most important semantic features of a user query, taking the motivation from [13]. This layer takes dimension-wise maximum value to form a final vector. This final vector is the embedding of a user utterance $e(u)$.

4.2 Utterance to Session Enc. Network

The structure of this network is very similar to that of the *Word to Utterance Encoder Network* with a key difference that there is no need of an Embedding layer. The embeddings of input utterances are taken as input. At each timestep, such an embedding is fed into the BiLSTM layer, which generates T hidden states. We pass these hidden states to a TDD Layer and further to a maxpool layer to get a max-pooled representation of the hidden states. In addition to this, we also use *skip connection* from BiLSTM layer to the softmax layer, for better flow of gradient. For this, we pass the last hidden state from BiLSTM layer h_T, and concatenate [9] it with the output of maxpool layer to get a final session embedding $e(s)$. This embedding captures appropriate signal from both the last utterance and the most significant utterances of the session.

4.3 Character to Word Encoder Network

We have found in our analysis [22] that presence of certain words in a sentence often leads to suitable classification in target class. However, if those words are not present in the sentence or are misspelled it often leads to wrong intent identification. Some of the often misspelled words from our system, which is currently live in our organization, are {"casal", "causal", "casaul", "casuaal"} for "casual", and {"montsh", "mnths", "monthhs"} for "months". In order to obtain the right representation for such misspelled words, we train the *Character Encoder Network* as shown in Figure 2.

This network takes sequence of one-hot encoding of the English alphabets which occur in a word. Using this network we predict the word embedding of correctly spelled words, as learned by using word2vec [19]. We keep two layers of BiLSTM followed by a time distributed dense and maxpool layer in this network. This network is very similar to *Word to*

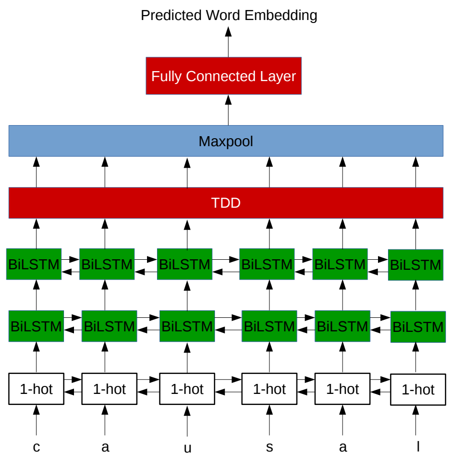

Figure 2: Character Encoder Network

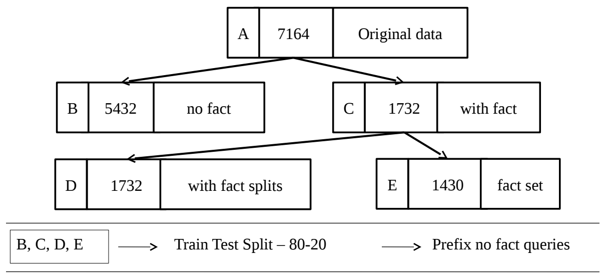

Figure 3: Data Preparation Process Details

Utterance Encoder Network. The loss function used is cosine similarity between the predicted word embedding and the actual word embedding (as learned using word2vec).

This network is used only for out of vocabulary words (w_{oov}) at the inference time to first predict the right word w_p from vocabulary as a replacement of the out of vocabulary words (w_{oov}). If the word is a misspelled word, the predicted word w_p is likely to be its correct replacement with the right spelling. If however, it is an unknown word, the predicted word will not be related to it. To avoid such mistakes, after obtaining the predicted word w_p, we also use *SequenceMatcher* [25] to find the similarity between the pair of words. It considers each word as a sequence of characters as shown in equation (2). It calculates a match score of the two words, if the score is above a certain threshold, we select the predicted word as the word to replace the OOV word, else we drop it.

$$sim(w_{oov}, w_p) = 2 * \frac{M}{T} \tag{2}$$

Where M is the number of matches and T is the total number of characters in the w_{oov} and w_p.

5 EXPERIMENTAL SETUP

5.1 Datasets

We used 3 datasets for carrying out the experiments :

Leave dataset : Consists of around 18k queries created in a manner as described in Section 5.2 later. This dataset belongs to leave management domain, hence we call it as leave dataset.

TCS Public Leave dataset : We also release a subset of Leave dataset in open source for the research community to carry out further experiments. This data has been split into fixed train, dev, test splits.

Yelp'13 : This dataset has been obtained from the Yelp Dataset Challenge in 2013 as used by Tang et al.[33]. The reviews have been rated from 1 to 5, where 1 denotes a poor review.

5.2 Data Preparation

We needed a dataset to conduct experiments as well as for usage as the seed model in actual production environment. This dataset should represent actual users' behavior, which is observed to contain pronoun references to entities mentioned in prior user utterances as well as to the facts pertinent to individual's situation, as shown in Table 1. We create such data from a dataset of completely self contained queries, currently in use in TCS.

Data Analysis: From the conversational assistant deployed in our organization, we extracted 7164 queries (see Figure 3), related to leave domain and analyzed this dataset manually. There were many different types of queries in this dataset. Some of these queries were completely self contained, and cannot be split in two parts, i.e., antecedent and query, for example, 'how many leave types are there?'. Most of the queries (5432) in this dataset were of this kind, since this system was designed for self contained queries only. Such queries are hereafter referred to as *no fact queries* (B). Second type of questions contain an antecedent clause along with a query, for example, "I am a Business Associate, can I avail a casual leave ?" has two clauses separated by a comma. Number of antecedent clause in such queries can sometimes be more than one. Similarly, sometimes the query comes before the fact. All such queries, which contain fact utterances before or after a query are hereafter referred to as *with fact queries*(C). We also observed that usually the number of such clauses were maximum five in the same query.

User utterance sequence preparation: First we manually split every *with fact query* into multiple different utterances, forming a sequence of utterances, referred to as *with fact splits*(D), as shown in Table 1. The distribution of length of such sequences is given in Figure 4. From such splits we

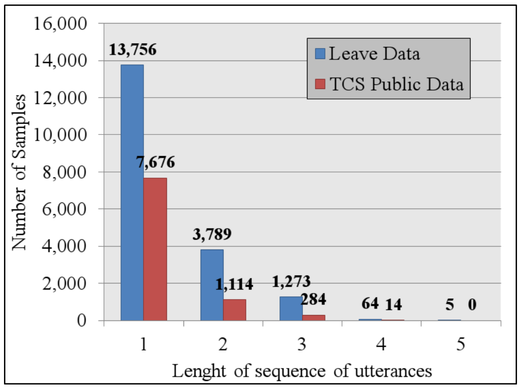

Figure 4: Distribution of Sequence Lengths

Property	Leave	TCS public data	Yelp'13
Train data	11031	5460	268,013
Val data	3677	1818	33,501
Test data	3677	1818	33,504
No. of classes	311	229	5

Table 2: Data Description

create another set of user utterances which contain only the facts and no queries, i.e., only first utterance is taken from this example. This set of utterances is hereafter referred to as *fact set*(E).

To create our final dataset, comprising of sequences of utterances x_i, and its intent y_i, we took all the B, C, D, and E datasets, as shown in Figure 3. The intent for all the queries in B, C and D datasets came from the original dataset, and for all the x_i of E dataset, we assumed a fixed intent y_f. We divided the queries of B, C, D and E datasets in two parts randomly in 80-20 ratio for Train and Test splits, then prepared the final dataset for each of these separately.

We created sequences of utterances of length 5, by prefixing *no fact queries* (taken from B dataset) to every query of the datasets B, C, D and E, to obtain B', C', D' and E' datasets. The data (union of B, C, D, E, B', C', D' and E') thus obtained was then manually reviewed to ensure that the intent for any of thus generated sequence of utterances appears right to a human being, other queries were dropped. This process gave us the data as shown in Table 2.

Pre-process data and Learn Word Embedding: After basic pre-processing steps such as lower case conversion, removal of special characters and abbreviation replacement we pass the data for learning word representations. (There was no stop-words removal step). We learn the word2vec [19] using skip gram algorithm. All the policy documents, chatbot answers as well as questions of all the query sets were used for learning these domain specific vector representations of all words. General purpose GLOVE word embeddings [24] learned on the English Wikipedia data was also tried, however it was observed that domain specific word embeddings render better accuracy, which could happen due to orthogonal meanings of the words such as "Leave".

5.3 Baselines Approaches

We compare our model *Deep Hierarchical Maxpool Network* (DHMN) with four other approaches, and demonstrate the efficacy of our approach in view of the chosen datasets.

Advanced Intent Model(AIM)[27]: As described in Section 1, we have deployed a system in TCS, which is being used by thousands of its employees. Details of the neural network architecture of this deployment have been given in [27], which is very similar to *Word to Utterance Encoder Network* as described in Section 4.1. To compare the efficacy of this algorithm on the data of this paper, we concatenate all utterances of a sequence of utterances into one sentence, before training and testing this model.

Hierarchical Attention Network(HAN) : HAN or hierarchical attention network [40], is the motivating approach for our DHMN, it uses a hierarchical model consisting of GRU at word level and at sentence level. They use attention at both the levels, to classify a document. We use the same algorithm for Abstract Anaphoric Intent Identification problem, and report the result in Section 6.

Hierarchical Maxpool Network(HMN) : HMN [40] uses a hierarchical model consisting of a combination of BiLSTM followed by a maxpool layer both at word level and at sentence level. We use two variants of this network, one with GRU as used in [40] and another with LSTM. This helps us study the contribution of LSTM (as compared to GRU) and that of last state of BiLSTM, taken as additional input into softmax layer in our variant of the network.

Hierarchical Character Network(HCN) : It is similar to HMN, with a difference that it takes sequence of characters as input instead of sequence of words in the first network (*Word to Utterance Network*). This type of representation of queries do not suffer from out of vocabulary words due to spelling mistakes, and are able to learn word variations. They however, lack the knowledge of semantic relatedness between words, which are captured by pre-trained word embeddings.

We do not use linear methods like Bag of Words [2, 7] or TfIdf [28] as a baseline, because these do not form a strong baseline against the above mentioned state of the art methods. However, we did carry out experiments using TfIdf, and found that they work only on self contained queries only, while with dataset requiring abstract anaphoric resolution, these techniques don't perform well.

5.4 Model Training

We compare our proposed algorithm DHMN on the four baseline algorithms, which we have described below. We create three splits of our datasets (Leave, TCS Public data) 60% for training, 20% each for validation and testing data. For Yelp'13 the dataset has been provided with fixed train, validation and test split. We performed hyper-parameter tuning for hierarchical maxpool network for the hidden number of nodes {100,150,..,350} with a step-size of 50 units, batch-size in range {16, 32, 64, 128}, and the learning rate in range {0.1, 0.01, 0.001, 0.0001}, and obtained the best set of parameters as chosen on the validation set. We obtained best results for number of hidden nodes as 300. Batch size is 32 on Leave dataset and 64 for yelp'13. The adam [12]optimizer gave the best results on all the datasets with a default learning rate of 0.001. We used Keras [3] to build these algorithms.

Regularization: LSTMs require a lot of training data and have huge number of parameters, as a result they tend to over-fit the training data easily, to prevent that we use techniques that have been proposed, including early stopping, L1/L2 regularization [20] (weight decay) and dropout. We used a dropout of 0.5 that worked well in this case.

6 MODEL EVALUATION RESULTS

We report average accuracy of various models, on the three datasets in Table 3. It can be observed that our proposed method outperforms the deployed algorithm (AIM) by a good margin of 6%. Since the deployed architecture does not use hierarchical structure, it has a disadvantage over the architectures which explore the hierarchical structure. We also proposed HCN as a benchmark, since it also uses a hierarchical structure but uses a character level representation. However, it does not perform well on Leave data, this could be because such a model lacks the capability of recognizing semantically similar words, which comes automatically from word2vec. Further, our model gives better performance than Hierarchical attention network by a margin of around 3% to 4%. Similarly, we use HMN from the same paper as a baseline algorithm and we find out that it performs better than HAN on our datasets, which is different from what Yang et al. [40] observed on their datasets. We achieved better results than HMN by a margin of about 2% using our model. We also ran experiments on HMN with LSTM and reported the results in the table. We perform a two tailed t-test to demonstrate the statistical significance of the results Deployed algorithm AIM, HAN and DHMN. We were able to reject the null hypothesis with a good margin on our datasets.

We also compare DHMN on a public dataset Yelp'13, with the baseline approaches. Yang et al. [40] reported that HAN performs better than HMN, while our variant of the HMN performs better than their benchmarks with a little margin. Hence, we conclude that our algorithm can achieve good results on large datasets, and generalizes well.

6.1 Model efficiency

We have studied the time taken to train our model and compared it with that of baseline approaches used in Table 4. We report the training time for each model, along with the number of trainable parameters. We found out that our model (DHMN) gets trained faster and also converges within a few epochs (below 10). Even though the number of trainable parameters are higher in HMN and DHMN, but it takes lesser time to train, probably due to maxpool layer. This is essential in production environment, whenever the model has to be retrained to incorporate new data.

6.2 Challenges in deployment

We have not yet deployed the DHMN algorithm in the production environment being used by real users. To deploy this in the production, there are certain challenges that need to be addressed. The deployed instance answers users' queries on about 15 different policies, e.g., Leave, Health Insurance, Working Hours policy etc. We have trained a different model for each of these policies using the AIM [27] model.

First, in this work we only looked into Leave data, and used its data to create the data for DHMN. However, it is quite a manual process to transform the data used for these 15 models into the format required by DHMN (as explained in Section 5.2. Hence, we need to automate this procedure to reduce the time and effort.

Second, every query to the system in production has to pass through a *high level intent classifier* which helps us choose one of the 15 models (Leave, Health Insurance etc.), and we then pass the query to the identified model. For deployment of DHMN, this high level intent identifier also needs to be changed to work on sequence of user utterances. Sometimes a sequence of user utterances may contain utterances belonging to more than one domain, i.e., Leave, Health Insurance etc. Therefore the data generation approach described in Section 5.2 will also need modification. Most of these don't appear to become a roadblock and we should be able to deploy the approach described in the current paper, in the production environment in coming months.

7 DISCUSSION AND CONCLUSION

We described a problem of *Abstract Anaphoric Intent Identification*, observed in an actual deployment environment of a conversational system. We also discussed at length why state of the art approaches present in research literature are either not applicable to this problem or cannot solve such a complex problem. We also described how hierarchical networks can be used to model this problem. This model is practically usable and can replace the current model, mainly because it outperforms the older model for self contained queries, and takes much lesser time to train. Using our little improvisation of this model, it outperformed the benchmarks on publicly available dataset and also performs better than various obvious baselines approaches on our dataset.

ACKNOWLEDGMENTS

The authors would like to thank TCS HR Leadership comprising of Mr. Ajoyendra Mukherjee, Dr. Ritu Anand, Ms. Shalini Gupta, and their teams such as Apoorva, Neelam, Rupali, Ankur etc.

REFERENCES

[1] Alexander Bartl and Gerasimos Spanakis. 2017. A retrieval-based dialogue system utilizing utterance and context embeddings. *CoRR* (2017).

	Algorithm	Leave	TCS Public data	Yelp'13
A	HCN	82.71	78.55	-
B	AIM[27]	86.84	80.62	-
C	HAN[40]	88.99	81.13	**68.20**
D	HMN-GRU[40]	90.78	83.15	66.90
E	HMN-LSTM	91.27	84.88	66.89*
F	**DHMN**	**92.47**	**85.23**	**68.42***

Table 3: Average accuracy (over 10 runs) comparison between baseline techniques and our algorithm DHMN. Numbers marked with * are averaged over 5 runs.

Algorithm	Params	Training time (s)
HCN	1,659,411	4563
AIM[27]	1,914,911	3060
HAN[40]	1,158,145	3102
HMN	3,537,611	2300
DHMN	2,639,711	1445

Table 4: Model efficiency: Parameters and training time per model

[2] William B. Cavnar and John M. Trenkle. 1994. N-Gram-Based Text Categorization. In *SDAIR-94, Annual Symposium on Document Analysis and Information Retrieval.*

[3] FranÃğois Chollet. 2015. keras.

[4] Zhou Chunting, Sun Chonglin, Liu Zhiyuan, and C. M. Lau Francis. 2015. A C-LSTM Neural Network for Text Classification. *arXiv:1511.08630* (2015).

[5] Kevin Clark and Christopher D. Manning. 2016. Deep Reinforcement Learning for Mention-Ranking Coreference Models. In *EMNLP*. The Association for Computational Linguistics, 2256–2262.

[6] Kevin Clark and Christopher D. Manning. 2016. Improving Coreference Resolution by Learning Entity-Level Distributed Representations. In *ACL (1)*. The Association for Computer Linguistics.

[7] Zellig Harris. 1954. Distributional structure. *Word* (1954).

[8] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.* (1997).

[9] J. Howard and S. Ruder. 2018. Fine-tuned Language Models for Text Classification. *ArXiv e-prints* (2018).

[10] Zongcheng Ji, Zhengdong Lu, and Hang Li. 2014. An Information Retrieval Approach to Short Text Conversation. (2014).

[11] Prerna Khurana, Puneet Agarwal, Gautam Shroff, Lovekesh Vig, and Ashwin Srinivasan. 2017. Hybrid BiLSTM-Siamese Network for FAQ Assistance. In *Proceedings of the ACM on Conference on Information and Knowledge Management (CIKM '17).*

[12] Diederik P. Kingma and Ba Jimmy. 2014. Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations (ICLR)* (2014).

[13] Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent Convolutional Neural Networks for Text Classification. In *AAAI Conference on Artificial Intelligence (AAAI).*

[14] Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. 2015. A Hierarchical Neural Autoencoder for Paragraphs and Documents.. In *ACL (1)*. The Association for Computer Linguistics.

[15] Jiwei Li, Alexander H Miller, Sumit Chopra, MarcÁurelio Ranzato, and Jason Weston. 2017. Learning Through Dialogue Interactions *(ICLR).*

[16] Rui Lin, Shujie Liu, Muyun Yang, Mu Li, Ming Zhou, and Sheng Li. 2015. Hierarchical Recurrent Neural Network for Document Modeling.. In *EMNLP*. The Association for Computational Linguistics.

[17] Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. 2015. The Ubuntu Dialogue Corpus: A Large Dataset for Research in Unstructured Multi-Turn Dialogue Systems. In *SIGDIAL Conference.*

[18] Ana Marasovic, Leo Born, Juri Opitz, and Anette Frank. 2017. A Mention-Ranking Model for Abstract Anaphora Resolution. In *EMNLP*. Association for Computational Linguistics.

[19] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems.*

[20] A.Y. Ng. 2004. Feature selection, l1 vs. l2 regularization, and rotational invariance. In *Proceedings of the twenty-first international conference on Machine learning*. ACM.

[21] Nikolaos Pappas and Andrei Popescu-Belis. 2017. Multilingual Hierarchical Attention Networks for Document Classification. (2017).

[22] Mayur Patidar, Puneet Agarwal, Lovekesh Vig, and Gautam Shroff. 2017. Correcting Linguistic Training Bias in an FAQ-bot using LSTM-VAE. In *DMNLP Workshop of ECML-PKDD.*

[23] Zhou Peng, Qi Zhenyu, Zheng Suncong, Xu Jiaming, Bao Hongyun, and Xu Bo. 2016. Text Classification Improved by Integrating Bidirectional LSTM with Two-dimensional Max Pooling. *Intl. Conf. on Computational Linguistics(COLING)* (2016).

[24] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP).*

[25] John W Ratcliff and David E Metzener. 1988. Pattern-matching-the gestalt approach. *Dr Dobbs Journal* (1988).

[26] Alan Ritter, Colin Cherry, and William B. Dolan. 2011. Data-driven Response Generation in Social Media. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP '11).*

[27] P. Mahesh Singh, Puneet Agarwal, Ashish Chaudhary, Gautam Shroff, Prerna Kurana, Mayur Patidar, Vivek Bisht, Rachit Bansal, Prateek Sachan, and Rohit Kumar. 2018. Hybrid BiLSTM-Siamese Network for FAQ Assistance. In *Proceedings of the IEEE International Conference on Data Engineering (ICDE '18).*

[28] Karen Sparck Jones. 1988. Document Retrieval Systems. Chapter A Statistical Interpretation of Term Specificity and Its Application in Retrieval.

[29] Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in neural information processing systems (NIPS).*

[30] Huan Sun, Hao Ma, Wen-tau Yih, Chen-Tse Tsai, Jingjing Liu, and Ming-Wei Chang. 2015. Open domain question answering via semantic enrichment. In *International Conference on World Wide Web (WWW).*

[31] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems (NIPS).*

[32] Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks. (2015).

[33] Duyu Tang, Bing Qin, and Ting Liu. 2015. Document Modeling with Gated Recurrent Neural Network for Sentiment Classification.. In *EMNLP.*

[34] Sam Wiseman, Alexander M. Rush, Stuart M. Shieber, and Jason Weston. 2015. Learning Anaphoricity and Antecedent Ranking Features for Coreference Resolution. In *ACL (1)*. The Association for Computer Linguistics.

[35] Yu Wu, Wei Wu, Zhoujun Li, and Ming Zhou. 2016. Topic Augmented Neural Network for Short Text Conversation. *CoRR* (2016).

[36] Yu Wu, Wei Wu, Chen Xing, Ming Zhou, and Zhoujun Li. 2016. Sequential Matching Network: A New Architecture for Multi-turn Response Selection in Retrieval-based Chatbots.

[37] Zhang Xiang, Jake Zhao Junbo, and LeCun Yann. 2015. Character-level Convolutional Networks for Text Classification. *Advances in Neural Information Processing Systems*.

[38] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. 2015. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. *CoRR* (2015).

[39] Mohit Yadav, Lovekesh Vig, and Gautam Shroff. 2017. Learning and Knowledge Transfer with Memory Networks for Machine Comprehension. In *Conference of the European Chapter of the Association for Computational Linguistics (EACL)*.

[40] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alexander J. Smola, and Eduard H. Hovy. 2016. Hierarchical Attention Networks for Document Classification. In *HLT-NAACL*.

[41] Xiao Yijun and Cho Kyunghyun. 2016. Efficient Character-level Document Classification by Combining Convolution and Recurrent Layers. *arXiv:1602.00367* (2016).

[42] Kim Yoon. 2014. Convolutional Neural Networks for Sentence Classification, In Empirical Methods in Natural Language Processing. *EMNLP*.

[43] Han Zhao, Zhengdong Lu, and Pascal Poupart. 2015. Self-Adaptive Hierarchical Sentence Model. (2015). http://arxiv.org/abs/1504.05070

[44] Xiangyang Zhou, Daxiang Dong, Hua Wu, Shiqi Zhao, Dianhai Yu, Hao Tian, Xuan Liu, and Rui Yan. 2016. Multi-view Response Selection for Human-Computer Conversation. In *EMNLP*.