Long-Term Context Awareness in a Conversational Agent: a Recurrent Neural Approach

Juan M. Huerta
Dow Jones & Co. Inc.
New York, NY 10036
juan.huerta@dowjones.com

Abstract

In this paper we describe Maxwell, our text-based multi-party conversational agent that delivers contextual knowledge derived from a corpus of 20 million news articles. We specifically focus on the context-monitoring component. Maxwell constantly tracks and models shift in conversational context by identifying topic breakpoints in conversational chains. Using a NARX recurrent network with reduced dimensionality CBOW word embedding features to model context shifts, we demonstrate an AUC of 0.701 in a sentence-based news domain topic shift task, which we consider encouraging initial results. We provide a brief description of Maxwell's architecture and approach and describe how we apply our context-awareness strategy in Maxwell.

1 Introduction

Task-oriented conversational agents typically combine dialog management functionality [3] with language understanding (or parsing), and possibly an ASR front-end and a natural language generation component. In these systems, there is typically a well-defined goal (or set of goals) and the agent's mission is to broker interactions with the end goal of furthering progress in terms of task achievement (e.g., [4]). While traditional assumptions are that there is an exclusive one-on-one interaction between the user and the agent, some work has been done around groups of users and thus multiparty engagements [16].

In this work we focus on Maxwell, which is our text-based conversational chat-bot for slack[1] whose goal, rather achieving a particular *transactional* task, is to provide relevant information contained in a very large set of newspaper articles (the knowledge base) at the relevant moment during multi-human natural chat conversations. Therefore, instead of operating in the traditional *direct* versus *mixed initiative* modalities, Maxwell works in *background mode*, passively listening to the conversation most of the time and only intervening (or barging-in) at moments when pieces of information (i.e., facts, articles) that are relevant to the conversation at some point exist in the knowledge base delivering these. In this paper we focus on Maxwell ability to track contexts (i.e., topics, entities, themes, and facts), which is used to detect topic breakpoints that trigger backend queries.

We specifically explain our approach that is based on Recurrent Neural Networks (RNN), specifically using a NARX network (Non-linear autoregressive with exogenous inputs) with CBOW word embeddings. In the next section we describe Maxwell's architecture and components, followed by our approach to context tracking and our experimental setup and results, finalizing with conclusions and thoughts regarding future work.

[1] https://api.slack.com/bot-users

2 Maxwell: a Knowledge-Oriented Conversational Agent

Maxwell is a text oriented bot with a conversational front end providing access to a large knowledge graph constructed around a very large set of news articles. The goal of Maxwell is to provide information during the course of human-to-human interactions (as opposed to carrying out transactional tasks). Maxwell is architected in 3 layers: the data-processing layer (Maxwell Pipeline), the Real-time summarization engine (accessible to applications through an API), and a suite of applications that access the summarization engine through the API. The particular end-user application we describe here is the conversational front end built as a slackbot. Figure 1 shows the architectural diagram of the 3 layers of Maxwell.

For the conversational bot, during the course of a conversation, once a topic and an entity are established, a query is released to the summarization engine API. The summarization engine narrows down on the shard most relevant to the query's entity and proceeds to analyze the shard using the query's context, returning a scored graph structure that the slackbot further evaluates, process and renders in brief textual form, it deems it adequate. Sharding is necessary because it is impractical to attempt to load the whole corpus into memory. We describe in more detail each of these modules.

2.1 Maxwell Large Scale Data-Ingestion Pipeline

Maxwell's backend is responsible for processing the news articles corpus; it runs in batch off line fashion. It is implemented in a parallelized way, specifically in Hadoop Map Reduce in Amazon elastic cloud and is thus capable of scale to handle an arbitrarily large corpus of news articles. Currently we have over 20 million English pieces (including articles, newswires, press releases, etc.) comprising clearly over 1 billion words; these articles were published through 6 months (contained in our Factiva database). Maxwell's pipeline ingests, annotates, summarizes, collates and indexes content. The result is a very large graph (the Maxwell Domain Graph), which is sharded by entities (people and companies, currently). Each article is consumed in parallel by the map tasks and XML-processed, parsed, and keyed by entity; in the reduce step each set of records is collated and further organized into shards. For the 6-month Factiva dataset we created a final custom on-disk tree-hierarchical structure in which nodes in the tree contain entity specific graphs while the tree itself is a *trie* based on entity-ID hashes. During query time, given a particular entity, it is very efficient to load into memory the corresponding entity-specific shard containing the pertaining structure and proceed with the summarization analysis.

2.2 Maxwell Dynamic Summarization Engine

The real time dynamic summarization engine receives the queries coming from the user-facing applications (through a set of RESTful API endpoints [2]) and based on the central entity in the query loads the relevant Maxwell Domain Graph shard and analyzes/produces a summarization object. A summarization object is a scored graph in which the nodes are articles, and edges are relations. There are 3 types of summaries or graphs that can be produced (depending on the endpoint accessed): linear (time oriented), graph (topic oriented) and tree (which is a minimum spanning tree of the topics). In each of the cases the nodes contain a score that reflects the relevance of the article (node) to the query. In the case of the graph and the tree structures, the edges represent document similarities above a certain threshold from the point of view of the context provided in the query (more below).

These 3 types of structures provide the user-facing application the ability to decide which structure to traverse, and how to do so, in order to find interconnections and prioritizations in the *rendering* of the final output. Depending on the application, a particular structure might be more advantageous (i.e., time-oriented traversals are best using the linear representation, the graph is best to identify communities of topics, while the tree (which is a minimum spanning tree) allows efficient distant node traversals). Each of these 3 types of graph analysis provides the ability to render a particular type of summary to the end user app (more below).

To score similarity between the context of a query and a particular node, the DSE maps queries and summaries to CBOW vectors [6, 7, 8]. For scoring it uses a POS-filtered CBOW vectors and calculates similarity score between pre-computed CBOW sets generated from the article and the CBOW vector of the query

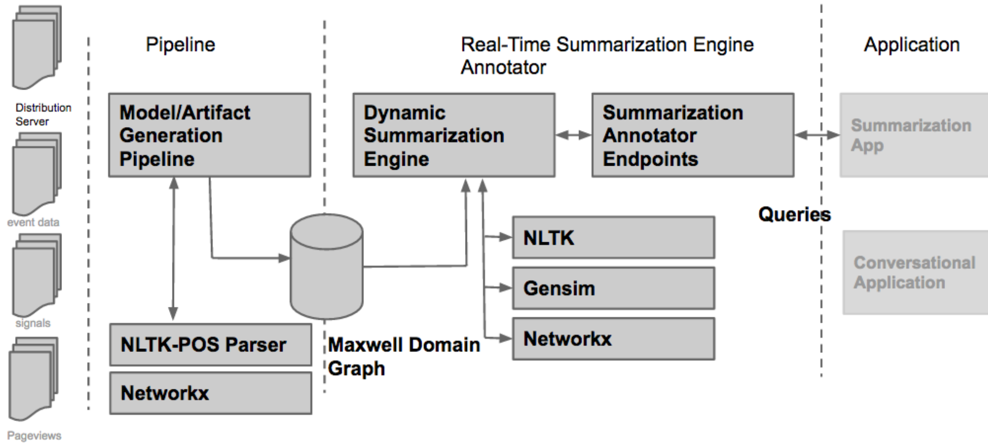

Figure 1. Schematic Diagram of Maxwell's Architecture

2.3 Maxwell Conversational Front-End

In the Maxwell architecture, the results delivered at the Summarization API endpoint by the DSE allow for the end-user interface (or front end) to be implemented in a variety of ways (visual interface, conversational, hybrid etc.). In this particular work we describe the conversational agent implementation. The conversational agent has a typical ([3]) architecture consisting of a Dialog manager (FSM), and a Language processing component, (which consists in turn of a Parsing, Entity extractor, POS-parsing and intention extraction), a backend-access point (the Maxwell DSE API) and a summary scoring and rendering component. It is purely text/visual based (not speech based).

In terms of Dialog Management, our conversational agent operates in one of 4 possible interaction modalities: (1) *direct* mode, i.e., being active mode in single-user conversations, (2) *asleep* mode, i.e., passive mode in single-user conversations, (3) *question-answering*, active mode in single-user 1 conversations in which each turn is assumed to be a self-contained question, and (4) *background* mode, i.e., multiparty passive conversations.

In our work we primarily focus on developing the multiparty passive dialog modality (the *background* mode), which requires Maxwell to be able to constantly listen a multi-party human conversation where topics and context are constantly changing, and barge-in and deliver a succinct point of view (or piece of information) when it's relevant. To support all 4 conversational modes, Maxwell needs to have combinations of dual context-awareness: short term (within-turn) for QA, directed and asleep modes and long-term (multi-turn) context-awareness for background mode. The background modality requires us to model context, implement a mechanism to calculate when to barge in, and to implement a result/summary-rendering component. In the rest of this paper, we focus on the long-term context-tracking feature of Maxwell and the approaches we are investigating.

3 Modeling Long-Term Context

In order to model long-term context, Maxwell addresses the problem as 2 sub-problems: (1) monitoring and identification of the topic and context breakpoints or significant shifts, (2) representation of the conversation segments as sets of keywords/key-phrases. We describe our approach to the first sub-problem: the identification of interaction breakpoints using a NARX network [13].

Let us assume that the set T represents a sequence of conversation turns ordered in chronological order $T=[t_1, t_2, \dots t_N]$ (these can be sentences or turns in a conversation). For each turn t_i we can generate the skipgram CBOW vector v_i representation (skip-gram with negative-sampling (SGNS), a word embedding method introduced by Mikolov et al. [6,7,8]

Using Google's word2vec in Gensim, specifically the GoogleNews-vectors-negative300 model, the dimensionality of the embedding vectors is 300. Based on $V=[v_1, v_2, \ldots v_N]$ we can generate $W=[w^1, w_2, w_3, \ldots]$ where each vector w_i is a 600 dimensional vector consisting of v_i and a concatenation of a vector d_i of time difference deltas where $d_{i,j}=(v_{i,j}-v_{i-1,j})$^2. In general, if the dimension of the CBOW model is d, the dimensionality of each vector in W is 2d.

Next we use the time series W and a vector of responses Y of length N (where every y_i is 0, except where there is a change in context, in which case y_i=1) to train a supervised classifier to recognize Y. If every vector w_i was an independent vector this would be a simple classification problem but as W represents a time series, we decided to apply a recurrent neural network to learn to identify shifts or changes in W. Specifically we trained a NARX network with 2d input nodes (where inputs are vectors w_i), d hidden nodes, 1 output node and output order = 2. This network takes multiple copies of the input (and possibly of intermediate layers). In our case it takes a copy of the vector w_(i-1) as input. The expectation is that the recurrent nature of the network will enable it to learn to identify changes, shifts and differences in the incoming multi-dimensional signal. NARX approaches have been applied in time series prediction (e.g., [13]), in this case our task is breakpoint identification. We use PyNeurGen[2].

In order to speed up the training process as well as to build more concise and parsimonious models, we implemented a simple dimensionality reduction process in which we select a random a subset of dimensions from the d original dimensions. Our original dimension d=300 and our target dimension d2=24. While there is a degradation in classification performance, as expected, this degradation is not too large to make this approach unusable, while increasing the speed of training.

4 Evaluation

In order to train and evaluate our context-tracking algorithm, we built a corpus consisting of the concatenation of the paragraphs contained in 2000 randomly selected news articles published in the first 6 months of 2015 in Factiva. From this article concatenation, we created a list containing one entry per each of the paragraphs of text in the articles. Each entry in this list corresponds to a paragraph in the article; the list is ordered and article boundaries are preserved. The task is to model the flow of textual language and identify the points in which the article boundaries are by detecting changes in topic/context. The total number of paragraphs in this corpus is 36,400. The average number of paragraphs per article is about 18. There total number of word tokens in the corpus is 1 million, and the average number of word tokens per paragraph is 27.5 with most paragraphs containing a couple of sentences per paragraph.

Thus, the list of paragraphs that conforms our corpus consists of 36,400 sample points. There are 2000 breakpoints contained in the 36,400 samples. Because this list is meant to represent a time series, we preserved the paragraph order of the list both in the training and eval phases of the task.

We split this list into two sublists of 60% and 40% length for training and eval purposes respectively. Therefore, there are about 1,200 topic breakpoints in the training set and 800 in the eval set. The task is then to process the eval time series and to produce a list of values (0 if we think there is no change in context and 1 if we thin there is); the input is consumed one turn at a time. Turns consist of 28 words on average per turn. On average, every 18 turns or so, a context shift occurs.

Using the training part of this corpus and preserving the order of the paragraph features in the set, we trained a NARX network using different values of target dimensions d2. Testing on the eval portion of the corpus we obtained the results shown in figure 2 below. In the left panel we show the ROC plot for the classifier trained on 24 dimensions. The area under the

[2] http://pyneurgen.sourceforge.net/

curve for this dimensionality is 0.701. In the right panel we show the Cumulative Gains Chart at a each of the 10 decile points. The vertical axis shows the percentage of positive responses. Each curve represents true positive response as a function of percentile at a specific dimension. The 3 curves reflect 8, 12 and 24 dimensions respectively from top to bottom. As we can see, higher dimensions increase performance. We observed that higher dimensionality also significantly increase computational complexity.

From these experiments we conclude the feasibility of the proposed approach. We demonstrated that using NARX networks and treating the incremental flow of text as time series in which chunks of 2 sentences are processed and analyzed for change in topic or context is a usable approach.

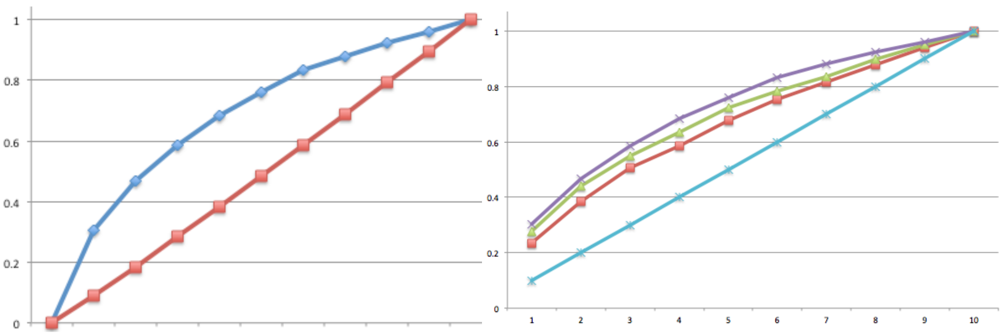

Figure 2. Time series context switch detection results: (a) ROC and (b) cummulative gains charts

5 Related and Relevant Work

From the point of view of summary generation, Rush et al [12] describe a method to generate summary content from observed article content. Their approach is based on a neural attention model, which can be customized using several encoder strategies. Their approach focuses on learning to produce headlines as a way to summarize content. We believe that this technique could be incorporated to Maxwell's output natural language generation component. Silber [14] and Yeh [17] each propose strategies to solve the same problem. Their techniques are based on less computationally demanding approaches, and could still be of use for our summarization..

6 Conclusions and Directions for Future Work

We have described in this paper a conversational agent capable of providing information relevant in a conversation based on a very large article base. We think that Maxwell can be improved across practically every constituent component: we believe we could explore new and improved dialog management strategies, different content summarization strategies, and leverage advances in question answering (e.g., [1, 5, 9, 15]) as well as knowledge base representation approaches [10, 11].

In addition to describing Maxwell's architecture, in this paper we have focused on the context-tracking algorithm we developed. We have obtained initial promising results based on a recursive neural network approach where the embedding vector is used as a time-varying signal. We have observed that our algorithm is robust when the text to be analyzed is similar to news article language. One possible direction for future work is to make our context-tracking algorithm more robust to human-to-human casual interactions and conversational language.

References

[1] Antoine Bordes, Nicolas Usunier, Sumit Chopra, Jason Weston. Large-scale Simple Question Answering with Memory Networks. arXiv Pre-Print, 2015

[2] Huerta, J. and Childs, C. “Accelerating News Integration in Automatic Knowledge Extraction Ecosystems: an API-first Outlook API’s” arXiv:1509.02783

[3] Huerta Pieraccini, “Where do we go from here? Research and Commercial Spoken Dialog Systems”, 6th SIGdial Workshop on Discourse and Dialogue 2005

[4] Mesnil et al., Using Recurrent Neural Networks for Slot Filling in Spoken Language Understanding. 2013

[5] Karl Moritz Hermann et. al. Teaching Machines to Read and Comprehend. arXiv Pre-Print, 2015.

[6] T. Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. In Proceedings of Workshop at ICLR, 2013.

[7] Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed Representations of Words and Phrases and their Compositionality. In Proceedings of NIPS, 2013.

[8] Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic Regularities in Continuous Space Word Representations. In Proceedings of NAACL HLT, 2013.

[9] Ndapandula Nakashole, Micro Reading with Priors: Towards Second Generation Machine Readers, ABC2014

[10] Arvind Neelakantan, Benjamin Roth and Andrew Mccallum. Knowledge Base Completion using Compositional Vector Space Models AKBC 2014

[11] Jay Pujara and Lise Getoor. Building Dynamic Knowledge Graphs AKBC 2014

[12] Rush et al, A Neural Attention Model for Abstractive Sentence Summarization, arXiv:1509.00685

[13] Sámek, David, and David Manas. "Artificial neural networks in artificial time series prediction benchmark." International Journal of Mathematical Models and Methods in Applied Sciences 5.6 (2011).

[14] Silber, H. Gregory, and Kathleen F. McCoy. "Efficient text summarization using lexical chains." Proceedings of the 5th international conference on Intelligent user interfaces. ACM, 2000.

[15] Tomasz Tylenda, Sarath Kumar Kondreddi and Gerhard Weikum. Spotting Knowledge Base Facts in Web Texts AKBC 2014

[16] David Traum and Stacy Marsella and Jonathan Gratch and Jina Lee and Arno Hartholt, “multi-issue, multi-strategy negotiation for multi-modal virtual agents” in Proc. of IVA, 2008

[17] Yeh, Jen-Yuan, et al. "Text summarization using a trainable summarizer and latent semantic analysis." Information processing & management 41.1 (2005).