Towards a Data-Driven Automation of Customer Support

Ricardo Rei
Instituto Superior Técnico
Lisboa
ricardo.rei@tecnico.ulisboa.pt

Luísa Coheur
Instituto Superior Técnico
Lisboa
luisa.coheur@l2f.inesc-id.pt

João Graça
Unbabel
Lisboa
joao@unbabel.com

Abstract

Customer support is essential for user satisfaction and critical for business. Its automation is a current research challenge. Yet, due to a lack of available data regarding this scenario, academic research has been mainly focused on community support, which is different from the typical customer support, where human agents are paid to give support to clients under a standardized language style. In this work, we analyse how retrieval- and generative-based deep learning models behave in a customer support scenario via email. Also, in an attempt to unify both approaches, we develop a simple hybrid model and a Generative Adversarial Network to jointly train both approaches. Our results suggest that a generative approach, based on a Sequence-to-Sequence model with attention, might be more suitable than a more complex retrieval-based approach. In addition, our human evaluation suggests that combining generation with retrieved answers (our hybrid approach), can improve the final answer quality.

1 Introduction

Customer support is laborious and time-consuming, and its automation is currently a hot topic in Natural Language Processing (NLP) (Lu et al., 2017; Hardalov et al., 2018), drawing both academic and industrial interest. If a corpus with interactions with the client is available, a data-driven approach is possible. The usual approach is either retrieval- or generative-based. In the former, the system has a repository of pre-defined answers that are used to reply to a given user request. In contrast, in the latter, every time a question is posed to the system, an answer is generated from scratch. Retrieval-based approaches are usually trained end-to-end with Dual Encoder Neural Networks (Lowe et al., 2015; Lu et al., 2017; Wu et al., 2017a; Yoon et al., 2017); on the other hand, generative-based approaches take advantage of Sequence-to-Sequence (Seq2Seq) models (Vinyals and Le, 2015; Boyanov et al., 2017; Hardalov et al., 2018). However, it is not clear which approach lead to better results. It is a common belief that generative models do not work well for a real setup, as they tend to produce generic or ungrammatical answers (Wu et al., 2017b). Nevertheless, Lowe et al. (2017) compared the

Preprint. Work in progress.

performance of a retrieval-based approach with two generative models[1] and concluded that answers produced by generative approaches are semantically closer to the ground-truth, and, in addition, generative models are better at staying on topic. Still, the corpus used in these experiments is the Ubuntu Dialog Corpus (Lowe et al., 2015), representing a scenario of community support, which is different from a typical customer support scenario, where human agents, engaged by a company, perform the support to their clients. In the latter scenario, template answers are commonly employed and the language used by the human agents is uniform. In contrast, in a community support scenario, the linguistic style between users is significantly different and answers to the same problems can be very diverse. So, a question arises: which is the approach to adopt in a typical customer support scenario, the retrieval- or the generative-based?

In this work we perform such study and we try to unite these two approaches with an hybrid model that uses the generated answer as input query for a retrieval system and a Generative Adversarial Network (GAN) that jointly trains these systems. Also, since we developed this work in collaboration with Unbabel and, traditionally, Unbabel focus has been in translating customer support messages via email, we will apply the described techniques into the email domain that, to the best of our knowledge, due to a lack of publicly available data, has not been properly explored.

This paper is organized as follows: in Section 2 we present the related work; in Section 3 we present the corpora to be used: Section 4 focus on the developed models; Section 5 describes our experiments and results; Finally, in Section 6 we show the main conclusions and point to future work.

2 Related Work

Bickel and Scheffer (2004) formulated the **Automatic Email Answering** problem as a mapping function between client emails and possible agent answers. These answers can either solve problems directly or provide pointers to support materials.

In more traditional NLP approaches the described mapping function can be modeled as a set of manually crafted text patterns (Sneiders, 2010; Sneiders et al., 2016) or as a multi-label classifier that, in a retrieval-based fashion, uses template answers to reply to new emails (Bickel and Scheffer, 2004; Yang and Kwok, 2012; Mota et al., 2013; Bonatti et al., 2016).

With the recent trend of neural networks, this function can be modeled with a Dual Encoder Neural Network in a retrieval-based approach (Lowe et al., 2015; Lu et al., 2017; Wu et al., 2017a; Yoon et al., 2017) or with a Seq2Seq model (Vinyals and Le, 2015; Boyanov et al., 2017; Hardalov et al., 2018) in a generative-based approach. Although, to the best of our knowledge, this type of deep learning models have never been used to model such function with email data, it has been successfully used for chat platforms (Lowe et al., 2015; Hardalov et al., 2018). However, comparisons between both approaches are rare, specially in the customer support domain.

Lowe et al. (2017) compared the performance of two generative models (an Hierarchical Recurrent Encoder-Decoder (Sordoni et al., 2015) and a LSTM based encoder-decoder) with a LSTM Dual Encoder (Lowe et al., 2015) using the embedding distance metrics, such as the *embeddings average*, *greedy matching* and *vector extrema*. After looking into these metrics the authors concluded that answers produced by generative models are semantically closer to the ground-truth, and that these models are better at staying on topic. However, based on another study from the same research group Liu et al. (2016), the authors concluded that metrics based on embedding distances do not correlate with human judgment. Also, since the Ubuntu Dialog Corpus is a community support corpus, conclusions from this study would be hard to draw for a customer support scenario.

Hardalov et al. (2018) also compared two generative models with a retrieval-based one using tweets from Apple customer support. In the referred work, the authors compared the performance of a LSTM-based Seq2Seq model and a transformer (Vaswani et al., 2017) from the OpenNMT toolkit (Klein et al., 2017) with a simple Term Frequency–Inverse Document Frequency (TF-IDF) based retrieval system. By looking to word-overlapping metrics and embedding distance metrics, the authors concluded that a Seq2Seq model with attention is able to generate better quality answers than

[1] An Hierarchical Recurrent Encoder-Decoder (Sordoni et al., 2015) and a Long Short-Term Memory (LSTM) based encoder-decoder with a LSTM Dual Encoder (Lowe et al., 2015) using the embeddings distance between the ground-truth and the answer returned by the model

the other two. Nonetheless, we believe that the retrieval-based baseline used in these experiments is weak, making it hard to draw conclusions about what approach works best.

In similar scenarios, Wang et al. (2017) argued that the two described frameworks (generative- and retrieval-based) should be jointly trained in a mini-max game with a GAN. Their experiments demonstrate significant improvements (23.96% on Precision@5 and 15.50% on Mean average precision over strong baselines in 3 different datasets: Web Search, Item recommendation and Question-Answering).

3 Corpora

Our main goal is to study the feasibility of deep learning models applied to customer support via email. For that reason, we will use a private customer support email corpus provided by Unbabel. Nonetheless, in order to benchmarks our results with publicly available data we will also use the Twitter Customer Support Corpus from Kaggle and the well-studied community support Ubuntu Dialog Corpus v2.0.

3.1 Private Email Corpus

This corpus was collected during 2016 and contains 42k email/answer pairs exchanged between customer and support agent. For each email the following preprocessings were applied: lower-casing, names removed, links replaced with a special URL token, Unicode characters converted to UTF-8, text split into tokens (using NLTK[2] tokenizer). In addition, all pairs containing at least one sequence with more than 300 tokens were removed from the corpus. After preprocessing the email pairs we ended up with 39244 email/answer pairs that we split into train/validation/test according to the following ratios 90%, 5% and 5% respectively, leaving 35319 pairs for training, 1963 for validation and 1962 for testing.

With the future intent of experimenting Dual Encoder networks, all the extracted pairs were labeled as positive samples. Negative samples were created by pairing each customer-email with a randomly selected agent-answer that had less then 0.7 cosine similarity from the original answer in a TF-IDF feature space. The reason we selected only answers with less than 0.7 cosine similarity is because in our domain we have many similar answers, and by blindly classifying answers as negative pairs we will end up creating valid pairs with a negative label. For the validation and test sets we created, then, for each email, a set of 10 possible answers, including the ground truth. This is later used to test the models in a ranking task, as in Lowe et al. (2015).

Finally, with all the unique answers from our training set, we created 1000 clusters in a TF-IDF feature space using a K-Means++ algorithm. Then, for each cluster, we selected the document closer to the centroid and created a list of 1000 possible template answers to be used later by retrieval-based models. We use a value of K equal to 1000 because, in this way, we guarantee a good answer coverage and, at the same time, retrieval-based models are able to compute in a few seconds what is the best candidate answer to a given question.

3.2 Twitter Customer Support

The Twitter Customer Support Corpus from Kaggle[3] is a large corpus of tweets and replies that intend to encourage research in natural language understanding and conversational models applied to the customer support scenario. Following (Hardalov et al., 2018), we isolate Apple support tweets, filtered tweets that redirected customers to direct messages and divided the corpus into training, validation, and testing sets by keeping the older posts for training. Also, user ids are anonymized, links replaced by the URL token and the text split into tokens (using NLTK[4] twitter tokenizer). After this step we end up with 45844 context/answer pairs for training, 1581 for validation and another 1581 for testing. In addition, as describes in Section 3.1, negative samples were created with a ratio one-to-one. Finally, in order to have template answers for both our retrieval-based and hybrid

[2] https://www.nltk.org/

[3] https://www.kaggle.com/thoughtvector/customer-support-on-twitter/

[4] https://www.nltk.org/index.html

approaches, we cluster all training answers into 1000 clusters with the same process described for the private email corpus.

3.3 Ubuntu Dialog Corpus V2.0

The Ubuntu Dialog Corpus (Lowe et al., 2015) consists of approximately one million conversation threads about community support for Ubuntu-related problems. The corpus was collected from Ubuntu-related chat rooms where each chat room has a particular topic. Users typically post a question in these rooms and after a while, a more experienced user replies with a possible solution. The data comes fully tokenized, stemmed, and lemmatized and entities like names, locations, organizations, URLs, and system paths were replaced with special tokens, thus no pre-processing was needed. Also, the training set provided is composed of triplets in the form (context, candidate answer, label), such that, the context contains all previous utterances in the conversation up to the point in which the next answer is to be found, the candidate answer can be either the ground truth or a randomly sampled answer, and the label is 1 or 0 indicating if the answer is a ground truth or not, respectively.

In addition to the training set, the authors also provide a validation set and a test set, but these sets instead of having triplets as explained for the training set, are composed by series containing a context and 10 possible answers including the ground truth. During their experiments they evaluate their models in the ability of ranking those 10 candidate answers.

In our work, in order to be consistent between different corpora, we split the training set into 3 parts (99%/0.5/%0.5%) to create our own training/validation/test sets. With these sets instead of evaluating the ranking ability of our model between epochs we looked into its accuracy.

In the Appendix we display a set of examples that show the differences between the 3 described corpus. Note that, for the Ubuntu Dialog corpus, the authors only made available a fully pre-processed version. Also, for the Private Email corpus, we could not show a more representative sample because for longer messages it becomes easy to identify the company behind the data.

4 Proposed Models

In this section we will start by briefly describing our retrieval- and generative-based models (Section 4.1 and Section 4.2, respectively). Finally we describe two ways of combining these approaches: an hybrid model that uses the generated answer to retrieve a template (Section 4.3) and a GAN that jointly trains a Dual Encoder and a Seq2Seq model in a zero-sum game (Section 5.4).

4.1 Retrieval-based Models

As we have seen before, retrieval-based models have a set of template answers that they can use to reply to new questions. Formally, the input to a retrieval-based model is a query q (the conversation up to this point or a client email message) and a potential template answer a. The model output is a score that, ideally, tells you how good is the template answer a given q. To find the most suitable answer to a new query q these models score multiple answers and choose the one with the highest score.

A commonly used baseline in such scenario is to compute the cosine similarity between a query q and all candidate answers in a TF-IDF feature space, and, then, rank the candidate answers accordingly. The motivation here is that answers tend to share words with the respective queries for which they were created. Based on that assumption, TF-IDF features can be computed for each document by "weighting" how important a word is for a specific document. We will use the described approach as a baseline but, following recent trends in NLP, we will also develop several Dual Encoder models that will learn to score query/answer pairs by looking into positive and negative examples.

4.1.1 Siamese-Like Dual Encoder

Lu et al. (2017) proposed a very simple Dual Encoder model to distinguish between positive and negative pairs, and later to rank candidate answers. The proposed model consists of an LSTM that encodes the query and another LSTM that encodes the answer. Both vectors are then concatenated into a single vector and passed to an Multi-Layer Perceptron (MLP) with a single hidden layer with 3

hidden units. The output layer is a softmax function that will determine if the question matches the answer or not.

Our baseline will consist in a similar model, but after some experimentation with the corpora to be used we decided to use a bidirectional LSTM instead of an unidirectional one, and an MLP hidden layer of size 300 instead of 3. We also replaced the softmax output with a sigmoid function for simplicity.

This model was initialized with pre-trained GloVe[5] word embeddings which were frozen during the first training epochs and fine-tuned during the last epochs. The model was trained to minimize the binary cross-entropy loss function using Adam optimizer.

4.1.2 Dual Encoder with Dual Attention

This Dual Encoder model consists of two bidirectional LSTM with shared parameters that are used to encode the query and the answer by summing the forward and backward passes. The final hidden state of the query encoder is then used to compute a soft-attention (Luong et al., 2015) over the answer's hidden states. Then, this attention-weighted answer embedding is concatenated with the final query hidden state and passed through a feed-forward tanh layer to create a contextualized query representation. Symmetrically, the same process is applied to the answer and the two resulting vectors are concatenated and passed through a MLP with one hidden layer and a sigmoid output.

The model was trained to minimize the binary cross-entropy loss function using Adam optimizer and, similar to the Siamese-Like Dual Encoder, it was initialized with pre-trained GloVe word embeddings, which were frozen during the first training epochs and fine-tuned during the last epochs.

4.1.3 ELMo Dual Encoder

Similar to all the previously presented architectures, this model consists of two bidirectional LSTMs. The first encoder is used to encode the query by concatenating the forward and backward passes. Similar to the Dual Encoder with Dual Attention model, the final hidden state of the query is used to compute a soft-attention (Luong et al., 2015) over the answer, or vice-versa depending on the corpus[6]. Finally, all the computed embeddings, namely, the query embedding, the answer embedding, and the attention embedding are concatenated and passed to an MLP with two hidden layers and a final sigmoid output.

Yet, the important piece of the described architecture is the word embedding layer. In contrast to the previously describes models, this Dual Encoder uses the contextualized word embeddings from ELMo (Peters et al., 2018) concatenated with fixed GloVe embeddings. This will not only produce contextualized information as well as fixed information about input words.

4.2 Generative-based Model

Unlike retrieval-based models, generative models can generate answers they have never seen before. However, as we have seen before, it is commonly believed that generative models do not work well for a real setup, as they tend to produce generic or ungrammatical answers and, for that reason, retrieval-based approaches are preferable. Nonetheless, taking into consideration that, in a customer support scenario, the answer space is narrow, we believe that Seq2Seq models can obtain a good performance, even for long documents such as emails.

As a generative model we have implemented a Seq2Seq model with attention consisting of two LSTMs with shared embeddings: an encoder and a decoder. Both LSTM are composed of 2 stacked layers. For each decoding step, an weighted averaged of all the encoder's hidden states is calculated using Luong's attention mechanism (Luong et al., 2015) and concatenated to the current decoder hidden state in order to help the decoder stay on topic.

The described model is initialized with GloVe embeddings and trained to maximize the conditional probability of the target sequence given the source query. Once training is complete, the described model can produce answers by finding the most likely sequence of words given a query q. Yet,

[5] `https://nlp.stanford.edu/projects/glove/`

[6] For the Ubuntu Dialog Corpus and for the Twitter customer support corpus since the contexts are much longer then the answers, the attention is computed over the context and not over the answer.

during test time, since the answer space is intractable, computing the most likely sequence becomes impracticable. For that reason, beam search is usually used. In a beam search decoding, at each step, we keep track of the top k most likely partial answers until k candidate answers are fully generated. This approach will lead to a near-optimal solution with a reasonable computation effort.

4.3 Hybrid Model

Alfalahi et al. (2015) formulated the concept of a shadow answer, i.e., a bag-of-words containing the **expected terms** in the answer. In the referred work, the authors computed a co-occurrence matrix between the terms used in the training queries and the respective answers. Then, during test time, when a new query arrives, the query is transformed into a shadow answer and the shadow answer is used as input for a retrieval-based system that ranks all template answers. When decoding with a Seq2Seq model, we are computing **expected terms**, one by one, by conditioning not only in the query tokens but also in the previously decoded tokens. In this way, we can think of the generated answer as a structured version of the shadow answer. In our hybrid approach, after having an answer provided by the previously described Seq2Seq model, we transform it into a TF-IDF feature space and we retrieve from the training data the answer with the highest cosine similarity.

4.4 Generative Adversarial Networks

Deep generative models based on RNNs are typically trained by maximizing log-likelihood. During training time, when generating the next word in a sentence it is common to feed the previous word from the ground truth (teacher forcing), instead of the one generated by the model (which may be incorrect). While this is helpful in guiding the training, especially in an early stage, this technique may cause exposure bias – the model does not know how to recover from a wrongly predicted word. One way to tackle this difference between training and inference is by inputting the generated word instead, with a given probability. Another way is to create a tailor-made loss for the task and use it to score the sentence (e.g. using BLEU for machine translation). However, for tasks such as answer generation, these metrics may not be informative enough to guide the training of the model.

Following the approach of Yu et al. (2017) both previous components (Sections 4.2 and 4.1.2) are re-used here: the Dual Encoder working as a discriminator and the Seq2Seq model working as a generator. Both are pre-trained in their original tasks, and later trained iteratively, one epoch for the generator and another one for the discriminator. The discriminator should now distinguish between real answers and synthetic ones, while the generator should produce samples good enough to fool the discriminator. We decided to use the Dual Encoder Dual Attention model and not the ELMo Dual Encoder as discriminator because the former was much faster in terms of computations.

Also following the approach proposed in Yu et al. (2017), the generator was trained using the REINFORCE algorithm (Williams, 1992), to overcome the impossibility of differentiating discrete outputs. At first each sequence was generated deterministically through greedy search, and one single reward was provided for the whole sequence, based on the discriminator's score. Later a more sophisticated reward function was used: for each partial sequence, 3 possible full sequences were generated by sampling from the softmax output. The average reward of these samples is used as the current token's reward. This makes it possible to evaluate incomplete sequences and to provide different rewards to each token.

5 Experiments

In this section, we describe our evaluation setup (Section 5.1) and our quantitative and qualitative results for both retrieval, generative and hybrid approaches (Section 5.2 and Section 5.3, respectively). Finally, in Section 5.4, we describe our attempts to unify retrieval and generative approaches with GAN models.

5.1 Evaluation

In this section, we will start by presenting the ranking metrics that are commonly used to evaluate retrieval-based models and, then, we will explain two types of evaluation metrics, commonly used in

natural language generation tasks, that measure how close are model answers from the ground-truth. Finally, we will describe the criteria used for our human evaluation.

5.1.1 Ranking Metrics

Retrieval-based models are commonly evaluated in an answer ranking task. From the candidate answers, only one answer is the original one and the goal of the model is to rank the correct answer as high as possible. The evaluation metrics used for this setup are $recall_N@k$ and Mean Reciprocal Rank (MRR)$_N$. Intuitively the $recall_N@k$ metric tells, on average, how many times the model ranked the ground-truth answer above a certain threshold. In-addition the MRR$_N$ tells on average how high the correct answer is ranked among N candidates.

5.1.2 Word overlap Measures

Word overlap measures evaluate the number of words that appear simultaneous in the predicted answer and in the ground-truth. For this type of metrics, we will use the METEOR (Lavie and Agarwal, 2007) and the ROUGE-L (Lin, 2004).

METEOR starts by creating an alignment between the candidate and reference sentences and mapping each word in the candidate sentence with the corresponding word in the reference (if that word exists). The corresponding word can either be a stem or a synonym and, based on this alignment, the METEOR score is the harmonic mean of precision and recall between the candidate sentence and the ground truth.

Likewise, the ROUGE-L is a harmonic mean of precision and recall but instead of being computed over word alignments it is based on the longest common sub-string between the candidate sentence and the ground truth. The longest common sub-string is defined as the longest set of words that occurs in both sequences in the same order, but contrarily to an n-gram, they do not need to be contiguous.

5.1.3 Semantic Evaluation Measures

Since in NLP two completely different sentences can share the same semantics, we decided to explore metrics that take language variability into consideration and look into the semantic meaning of the answers provided. A commonly used alternative to word overlap measures, is to use methods based on word embeddings, such as *embedding average*, *greedy matching* and *vector extrema*.

The *embedding average* metric consists in creating sentence level embeddings by averaging the word embeddings in two different sentences and, in order to assess their similarity, compute the cosine similarity.

Similarly, the *vector extrema* computes sentence level embeddings from word embeddings but, instead of doing it by averaging all words in a sentence, the sentence level embedding is computed by taking the most extreme values along all dimensions. The intuition behind this metric is that more informative words will be highlighted in comparison to common words. This is due to a property of the embedding space in which common words (e.g. stopwords) tend to lie closer to the origin of the semantic space.

In contrast, the *greedy matching* greedily looks for the best possible alignment between the word embeddings of two sentences by minimizing their cosine similarity. The total score is then computed by the average across all words.

5.1.4 Human Evaluation

Since the previously described metrics do not correlate well with human judgment (Liu et al., 2016), we perform a human evaluation to compare the described approaches. We randomly selected 50 contexts from the Twitter Customer Support corpus test set and used our models to generate/retrieve an answer. The annotator is then asked to evaluate the quality of the three models, along with the ground truth, by providing a score from 1 (worse) to 5 (best) to each answer. The annotation criteria are: 1 – completely irrelevant answer; 2 – irrelevant answer; 3 – related but generic answer; 4 – relevant answer; 5 – perfect answer.

5.2 Quantitative analysis

Taking into consideration the evaluation metrics described in the previous section, we start by evaluating our retrieval-based models in an answer ranking task. Then, after selecting our best retrieval model, we compare it against our generative and hybrid models in an answer generation task using word overlapping metrics and embedding distance metrics. We decided to exclude the Ubuntu Dialog corpus from some experiments due to the fact that, this corpus, is a community support corpus rather than a customer support one and, for that reason, we had no template answers to use.

5.2.1 Answer Ranking Task

Tables 1, 2 and 3 summarize our results for the retrieval-based models in the described ranking task. In these tables, TF-IDF is our retrieval baseline, SLDE stands for Siamese-like Dual Encoder (Section 4.1.1), DADE stands for Dual Attention with Dual Encoder (Section 4.1.2) and ELMoDE is the Dual Encoder with ELMo embeddings (Section 4.1.3).

After observing that the ELMoDE model achieved superior performance across all corpora, and since both the Twitter corpus and the private email corpus are much smaller than Ubuntu Dialog corpus, we experimented with a transfer learning technique in which the ELMo Dual Encoder model is first trained with Ubuntu data and then adapted to a smaller corpus. The adapted model is referred as "ELMo+transfer" in Table 2 and 3. This adaptation techniques yield to improvements for both corpora. In addition we observed that starting the training with a previously trained model provides a much better initialization, that, in consequence, leads to a faster convergence and better local minimal (Figure 1).

Table 1: Ubuntu Dialog corpus ranking task results.

Ubuntu Dialog corpus					
	$R_{10}@1$	$R_{10}@2$	$R_{10}@3$	$R_{10}@5$	MRR
TF-IDF	0.5006	0.6210	0.6951	0.8127	0.6372
SLDE	0.3031	0.5052	0.6556	0.8479	0.5200
DADE	0.6060	0.7717	0.8595	0.9479	0.7461
ELMoDE	**0.6468**	**0.8095**	**0.8866**	**0.9590**	**0.7766**
Lowe et al. (2015)	0.5520	0.7210	-	0.9240	-

Table 2: Apple Customer Support on Twitter ranking task results.

Apple Customer Support on Twitter					
	$R_{10}@1$	$R_{10}@2$	$R_{10}@3$	$R_{10}@5$	MRR
TF-IDF	0.4035	0.5215	0.5993	0.7081	0.5478
SLDE	0.3438	0.5307	0.6768	0.8564	0.5476
DADE	0.4437	0.6278	0.7454	0.8738	0.6224
ELMoDE	0.5082	0.7090	0.8201	0.9235	0.6804
ELMoDE+transfer	**0.5708**	**0.7489**	**0.8441**	**0.9361**	**0.7220**

Table 3: Customer Support via Email ranking task results.

Customer support via Email ranking results.					
	$R_{10}@1$	$R_{10}@2$	$R_{10}@3$	$R_{10}@5$	MRR
TF-IDF	0.3730	0.5118	0.5949	0.7060	0.5283
SLDE	0.3037	0.5281	0.6892	0.8823	0.5307
DADE	0.4616	0.6927	0.8211	0.9325	0.6555
ELMoDE	0.5417	0.7656	0.8736	0.9592	0.7152
ELMoDE+transfer	**0.5669**	**0.7860**	**0.8831**	**0.9620**	**0.7327**

5.2.2 Answer Generation Task

Tables 4, 5 and 6 present the obtained results for the different corpora in an answer generation task in which the model answer $\hat{a}$ is compared against the ground truth answer a in terms of word overlapping and embedding distances metrics. In the referred tables, the generative model refers to the Seq2Seq model explained in Section 4.2 and the hybrid model refers to the Seq2Seq model plus answer retrieval explained in Section 4.3. In Table 5 and Table 6 we can also observe a retrieval model. The referred retrieval model is the ELMo Dual Encoder model explained in Section 4.1.3 and adapted from the Ubuntu corpus to these two corpora. For the Twitter corpus and for the private email corpus, due to a clear language standardization and template usage, we were able to extract 1000 answer templates and use those as candidate answers for the retrieval-based and hybrid models. In contrast, for the Ubuntu Dialog corpus, due to a significantly different linguistic style between users, we could not find a set of candidate answers that guaranteed a good answer coverage and fast computations. For that reason, we were not able to evaluate our retrieval-based model in such corpus. However, since it is relatively fast to compute distances inside a TF-IDF space, we were able to experiment with our hybrid model assuming that, in the retrieval phase, all the training answers are possible candidates.

Model	METEOR	ROUGE-L	Embeddings Average	Vector Extrema	Greedy Matching
generative (beam-1)	0.1835	0.2579	**0.5191**	**0.4045**	**0.7655**
generative (beam-3)	0.2478	0.3290	0.4104	0.3361	0.7376
generative (beam-5)	**0.2615**	**0.3382**	0.3692	0.3131	0.7271
hybrid (beam-1)	**0.2740**	0.2988	**0.4252**	**0.3529**	0.7259
hybrid (beam-3)	0.2715	**0.3341**	0.3663	0.3274	**0.7280**
hybrid (beam-5)	0.2697	0.3337	0.3589	0.3197	0.7247

Table 4: Answer generation results for the Ubuntu Dialog corpus v2.0.

Model	METEOR	ROUGE-L	Embeddings Average	Vector Extrema	Greedy Matching
generative (beam-1)	0.1678	0.2368	**0.8903**	0.4185	0.7508
generative (beam-3)	**0.2232**	**0.2675**	0.8823	**0.4693**	**0.7911**
generative (beam-5)	0.2217	0.2641	0.8790	0.4639	0.7878
hybrid (beam-1)	0.1104	0.1809	0.8733	**0.4786**	**0.7886**
hybrid (beam-3)	0.1291	**0.1941**	0.8812	0.4635	0.7809
hybrid (beam-5)	**0.1297**	0.1936	**0.8831**	0.4627	0.7793
retrieval (best)	**0.1298**	**0.1723**	**0.8600**	**0.4421**	**0.7505**

Table 5: Answer generation results for the Apple Twitter corpus.

Model	METEOR	ROUGE-L	Embeddings Average	Vector Extrema	Greedy Matching
generative (beam-1)	**0.2565**	**0.3614**	0.9073	**0.5034**	**0.7906**
generative (beam-3)	0.2241	0.2916	0.9382	0.4596	0.7809
generative (beam-5)	0.2149	0.2890	**0.9401**	0.4492	0.7811
hybrid (beam-1)	**0.2083**	**0.3127**	0.9262	**0.4849**	**0.7856**
hybrid (beam-3)	0.1681	0.2407	0.9381	0.4569	0.7711
hybrid (beam-5)	0.1566	0.2404	**0.9387**	0.4458	0.7807
retrieval (best)	**0.1710**	**0.2589**	**0.8532**	**0.4385**	**0.7371**

Table 6: Answer generation results for the private email corpus.

Analyzing the previous tables we can observe that our generative model seems to outperform all the others. Also, for the Twitter corpus and for the private email corpus, we can observe that our retrieval-based model is the one with the lowest scores both in terms of word overlapping metrics and embeddings distance. Despite that, although word-overlapping and embedding distance metrics have shown similar correlation with human judgment (Sharma et al., 2017), this correlation is weak and more reliable evaluation is required in order to decide which model works best in a customer support scenario.

5.3 Qualitative analysis

Considering that the usual metrics can lead to inaccurate results, as answers tend to share some phrases or even sentences, which are positively scored by these metrics, but are semantically different

(e.g: “We’d love to help out. Tell us more details about what you’re experiencing, and we’ll check out some options for you.” vs “We’d love to help out. Tell us more details of what you’re experiencing. Which device are you trying to update?”), we decided to perform a human evaluation. This evaluation was only done using the Twitter corpus. The private email corpus, due to privacy reasons, was excluded from this experiment, and the Ubuntu Dialog corpus was excluded for not being customer support but rather community support.

This evaluation was performed by 30 annotators, organized into groups of 6. Each group received a batch of 10 different contexts and 4 possible answers for each context: the ground-truth, the answer from the Seq2Seq model, the answer from the ELMoDE model and the answer from the hybrid model. The annotators received the criteria described in Section 5.1.4. Also, the great majority of annotators were Apple device users recruited from a MSc. in Information Management taking a text analytics course. Results are shown in Table 7.

	Human Score Distribution (%)							
Model	**1**	**2**	**3**	**4**	**5**	**Average Score**	**Fleiss Kappa**	95 % CI
Hybrid	28.67	17.0	21.33	21.67	11.33	2.7	0.14	[0.10, 0.19]
Generative	29.67	20.33	20.0	21.00	9.0	2.593	0.13	[0.09, 0.17]
Retrieval	30.33	22.0	23.67	17.0	7.0	2.483	0.08	[0.04, 0.13]
Ground Truth	7.33	10.67	20.67	30.0	31.33	3.273	0.08	[0.03, 0.13]

Table 7: Scores given by human annotators on the different models. CI stands for confidence interval.

In contrast to the obtained automatic results shown in Table 5, our hybrid model obtains a higher average score than our generative model. Although with a slight Fleiss agreement (Landis and Koch, 1977), these results, yet again, suggest that automatic evaluation for task-oriented dialog systems correlates weakly with human judgement (Sharma et al., 2017).

5.4 Generative Adversarial Networks

In an attempt to combine both previous approaches (retrieval and generative) a GAN was used to generate answers for the private email corpus. Both the generator and the discriminator were pretrained separately, and then iteratively trained in an adversarial setup in which the discriminator’s guess (of whether a sample was real or synthetic) is used as a signal to update the generator’s parameters. In our first experiments, a simple approach was used to design the generator’s reward function: one single reward was given to a whole answer after it was generated. This has the major drawback of not distinguishing which words were responsible for poor/good results. Although this approach may provide reasonable results in different contexts such as videogames (Karpathy, 2016), in this case a bizarre behaviour became dominant. After a few training iterations, the pre-trained generator started to output a single symbol repeatedly. In different runs it would get stuck on different symbols. This issue could be caused by exaggerated learning rates (although the validation loss never decreased even with much lower learning rates). Alternatively, the simplistic reward adopted could be the reason behind this.

To investigate this issue further a more elaborate reward was designed, following Yu et al. (2017). At each step of the sequence, a unique reward was provided to the partial sequence generated so far. To do this, 3 full answers were generated from each partial sentence, and the average reward provided by the discriminator would become the partial sentence’s reward. To escape the deterministic behaviour of our model, each token was sampled instead of always selecting the most likely one. This allowed to provide different reward values to each word of the sentence. The behaviour of picking one token repeatedly disappeared, but the validation loss increased even further than before. We hypothesize that this may be due to improper pre-training of the discriminator (either too much or too little could cause it to provide bad reward values). Poor hyper-parameter choice could also be the cause, since balancing the amount of training of the discriminator and generator may impact greatly results.

An interesting aspect of this model, which could not be explored due to this initial issue, is that the discriminator is not necessarily learning the same task during pre-training and training. In pre-training, the discriminator must distinguish between correct and incorrect question/answer pairs. During training time, it must distinguish between "real" answers and a generated one. If the discriminator is not pre-trained, it may have no incentive to look at the question when classifying the answer. It may also lose this behaviour during training, if the generated answers are not fluent enough.

6 Conclusions and Future Work

Taking into account the language standardization and template usage, typically observed in customer support scenarios, we show that a simple generative approach, based on a Seq2Seq model with attention obtains better results than a more complex retrieval-based approach, based on a dual encoder model.

In our future work we plan to study other customer support resources and access the performance of more complex Seq2Seq models. We also believe that hybrid models, such as the one proposed by Wu et al. (2018), that incorporates retrieved answers as start-point for generation, might work well for the customer support scenario.

References

Alfalahi, A., Eriksson, G., and Sneiders, E. (2015). Shadow answers as an intermediary in email answer retrieval. In Mothe, J., Savoy, J., Kamps, J., Pinel-Sauvagnat, K., Jones, G., San Juan, E., Capellato, L., and Ferro, N., editors, *Experimental IR Meets Multilinguality, Multimodality, and Interaction*, pages 209–214, Cham. Springer International Publishing.

Bickel, S. and Scheffer, T. (2004). Learning from message pairs for automatic email answering. In Boulicaut, J.-F., Esposito, F., Giannotti, F., and Pedreschi, D., editors, *Machine Learning: ECML 2004*, pages 87–98, Berlin, Heidelberg. Springer Berlin Heidelberg.

Bonatti, R., de Paula, A. G., Lamarca, V. S., and Cozman, F. G. (2016). Effect of part-of-speech and lemmatization filtering in email classification for automatic reply. In *Knowledge Extraction from Text, Papers from the 2016 AAAI Workshop, Phoenix, Arizona, USA, February 12, 2016.*

Boyanov, M., Nakov, P., Moschitti, A., Da San Martino, G., and Koychev, I. (2017). Building chatbots from forum data: Model selection using question answering metrics. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017*, pages 121–129. INCOMA Ltd.

Hardalov, M., Koychev, I., and Nakov, P. (2018). Towards automated customer support. *arXiv preprint arXiv:1809.00303*.

Karpathy, A. (2016). Deep reinforcement learning: Pong from pixels.

Klein, G., Kim, Y., Deng, Y., Senellart, J., and Rush, A. (2017). Opennmt: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72, Vancouver, Canada. Association for Computational Linguistics.

Landis, J. R. and Koch, G. G. (1977). The measurement of observer agreement for categorical data. volume 33, pages 159–174. Wiley, International Biometric Society.

Lavie, A. and Agarwal, A. (2007). Meteor: An automatic metric for mt evaluation with high levels of correlation with human judgments. In *Proceedings of the Second Workshop on Statistical Machine Translation*, StatMT '07, pages 228–231, Stroudsburg, PA, USA. Association for Computational Linguistics.

Lin, C.-Y. (2004). Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Liu, C.-W., Lowe, R., Serban, I., Noseworthy, M., Charlin, L., and Pineau, J. (2016). How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. pages 2122–2132.

Lowe, R., Pow, N., Serban, I., Charlin, L., Liu, C.-W., and Pineau, J. (2017). Training end-to-end dialogue systems with the ubuntu dialogue corpus. volume 8, pages 31–65.

Lowe, R., Pow, N., Serban, I., and Pineau, J. (2015). The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 285–294, Prague, Czech Republic. Association for Computational Linguistics.

Lu, Y., Keung, P., Zhang, S., Sun, J., and Bhardwaj, V. (2017). A practical approach to dialogue response generation in closed domains. *arXiv preprint arXiv:1703.09439*.

Luong, T., Pham, H., and Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Mota, P., Martins, B., and Coheur, L. (2013). Designing an answer template retrieval system. Technical Report 35, INESC-ID.

Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Sharma, S., Asri, L. E., Schulz, H., and Zumer, J. (2017). Relevance of unsupervised metrics in task-oriented dialogue for evaluating natural language generation. *arXiv preprint arXiv:1706.09799*.

Sneiders, E. (2010). Automated email answering by text pattern matching. In Loftsson, H., Rögnvaldsson, E., and Helgadóttir, S., editors, *Advances in Natural Language Processing*, pages 381–392, Berlin, Heidelberg. Springer Berlin Heidelberg.

Sneiders, E., Sjöbergh, J., and Alfalahi, A. (2016). Email answering by matching question and context-specific text patterns: Performance and error analysis. In Rocha, Á., Correia, A. M., Adeli, H., Reis, L. P., and Mendonça Teixeira, M., editors, *New Advances in Information Systems and Technologies*, pages 123–133, Cham. Springer International Publishing.

Sordoni, A., Bengio, Y., Vahabi, H., Lioma, C., Grue Simonsen, J., and Nie, J.-Y. (2015). A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, pages 553–562, New York, NY, USA. ACM.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Vinyals, O. and Le, Q. V. (2015). A neural conversational model. *arXiv preprint arXiv:1506.05869*.

Wang, J., Yu, L., Zhang, W., Gong, Y., Xu, Y., Wang, B., Zhang, P., and Zhang, D. (2017). IRGAN: A minimax game for unifying generative and discriminative information retrieval models. *CoRR*, abs/1705.10513.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. volume 8, pages 229–256, Hingham, MA, USA. Kluwer Academic Publishers.

Wu, Y., Wei, F., Huang, S., Li, Z., and Zhou, M. (2018). Response generation by context-aware prototype editing. *arXiv preprint arXiv:1806.07042*.

Wu, Y., Wu, W., Xing, C., Zhou, M., and Li, Z. (2017a). Sequential matching network: A new architecture for multi-turn response selection in retrieval-based chatbots. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 496–505. Association for Computational Linguistics.

Wu, Y., Wu, W., Xing, C., Zhou, M., and Li, Z. (2017b). Sequential matching network: A new architecture for multi-turn response selection in retrieval-based chatbots. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 496–505. Association for Computational Linguistics.

Yang, W. and Kwok, L. (2012). Improving the automatic email responding system for computer manufacturers via machine learning. pages 487–491.

Yoon, S., Shin, J., and Jung, K. (2017). Learning to rank question-answer pairs using hierarchical recurrent encoder with latent topic clustering. *CoRR*, abs/1710.03430.

Yu, L., Zhang, W., Wang, J., and Yu, Y. (2017). Seqgan: Sequence generative adversarial nets with policy gradient.

Appendices

Corpus	Context/Question	Answer
Private Email Corpus	It seems I set up my business account with the same email I used for an older account. Now when I log in it brings me to the old account and not the one I set up recently. I do not know how to access the new business account _URL_.	hi _NAME_ did you recently close your email account?
Apple Customer Support on Twitter	@AppleSupport, I have an iPhone 7, just did the update and now my phone keeps shutting off and re-booting itself every time I try to read or do anything on my phone! Frustrating just started this morning any suggestions! FYI my son having same issue both of us on a 7 @User Hi there! Did you both update to iOS 11.2? There are a few more steps to try, if so: URL @AppleSupport Yes we did!	_USERID_ Great! You may also need to disable notifications, restart your device, then re-enable those. More info here: _URL_
Ubuntu Dialog Corpus	hey guy , what be the standard kernel sourc call if i be gon na upgrad my kernel from the generic one that come with the livecd . best to use apt-get , or just go and get vanilla sourc from kernel.org ? also , be there anyth i need to know about compiling/instal a new kernel after use the livecd generic one (i be familiar with kernel compil under gentoo) __eou__ __eot__ well.. use apt be a lot easier.. but sure.. go ahead and compil your own if you want.. just keep the old one around.. just in case __eou__ they call me start.. you just have to live with it =) __eou__ __eot__ script a short loop ? __eou__ __eot__ a quick script __eou__ i just want to run a coupl of line of bash command everi second.. __eou__ __eot__	as someon suggest , either put both command in a text file (shell script) and run it from cron , or els write a short script to execut everi 2 second . slart , wahat be the command ?

Table 8: Customer context/question plus the following answers across the 3 different corpus.

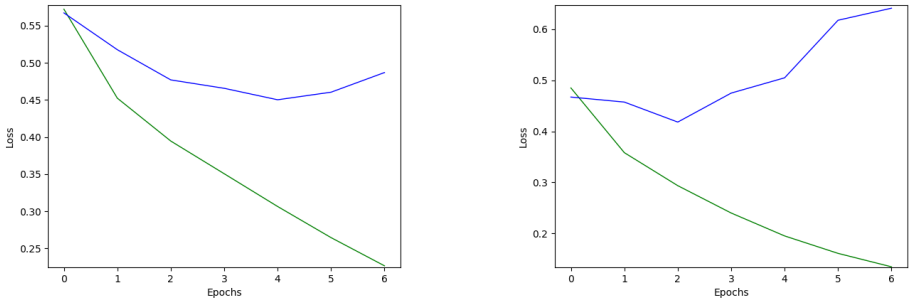

Figure 1: Loss curve comparison over epoch for the ELMo Dual Encoder model trained from scratch in Apple Twitter data (left side) and adapted from Ubuntu Dialog corpus to the Apple Twitter data (right side). In both figures the green line represents the training loss while the blue line represents the validation loss.