DIRECT OPTIMIZATION OF F-MEASURE FOR RETRIEVAL-BASED PERSONAL QUESTION ANSWERING

Rasool Fakoor, Amanjit Kainth, Siamak Shakeri*, Christopher Winestock, Abdel-rahman Mohamed, Ruhi Sarikaya*

Amazon

ABSTRACT

Recent advances in spoken language technologies and the introduction of many customer facing products, have given rise to a wide customer reliance on smart personal assistants for many of their daily tasks. In this paper, we present a system to reduce users' cognitive load by extending personal assistants with long-term personal memory where users can store and retrieve by voice, arbitrary pieces of information. The problem is framed as a neural retrieval based question answering system where answers are selected from previously stored user memories. We propose to directly optimize the end-to-end retrieval performance, measured by the F1-score, using reinforcement learning, leading to better performance our experimental test set(s).

Index Terms— Question Answering, Spoken information retrieval, Reinforcement Learning, Personal Assistants

1. INTRODUCTION

Recent advances in speech recognition [1, 2], natural language understanding [3, 4], question answering [5, 6, 7], and dialogue systems [8, 9] have fueled the current surge in research and development for smart personal assistants [10] like Alexa, Siri, Google assistant, and Cortana with may use cases around shopping, music, etc.

In this paper we present a system for providing personal assistants a long term personal memory that enable users to store anything they want to remember by voice, and then later ask questions about it. An example use case is shown in Table 1. This system extends long-term memories of users and enables them to store and retrieve arbitrary pieces of information they are juggling in their minds.

The system is framed as a question answering (QA) system over user generated memories which is related to QA systems with answers extracted from unstructured public sources like wikipedia [11], neural information retrieval [12], text matching [13], and machine comprehension [14].

One of the challenges for retrieval models is that they are trained on criteria other than the needed business metrics which are not ustually differentiable, e.g. train on pairwise matching while the overall performance is measure by $F1$ score. For example, [15] proposed a method for direct optimization of the relevance loss functions in ranking problems via structured estimation in Hilbert spaces by formulating it as a linear assignment problem.

* equal contribution

Question:
what did i do with ben's cell phone
Answers:
1. ✓ i gave benny's cell in for repairs at the store on first street
2. ✓ i left ben's iphone on the kitchen table
3. ✓ i sent bennie's old phone to mat
4. × ben wants a new cell phone for his birthday
5. × dad's cell is an iphone eight
6. × the screen of benjamin's phone is broken

Table 1. An of example QA group. These samples are output of an ASR system and hence normalized, i.e. noisy data, no punctuation, no capitalization, etc.

In [16] the authors proposed a method to directly optimize a relaxed version of the F-measure which is similar to a variant we present in this paper. In [17], expected F-measure is used to train a neural parsing model on sentence-level $F1$.

Another way to deal with non-differentiability in functions is to approximate the gradients using the REINFORCE algorithm [18]. In [19, 20], REINFORCE was used for sentence generation both in machine translation and video captioning tasks. The goal in both papers was to directly optimize evaluation metrics of interest such as BLEU-4 or CIDEr.

In this paper, we focus on improving the overall performance by incorporating the $F1$ score as part of the optimization objective. The main contributions of the paper are:

- Introducing a system for spoken personal QA.
- Proposing a method to directly optimize $F1$, and introducing stable optimization strategies.
- Present extensive empirical evidence and analysis discussing the viability of this approach, comparing it to traditional optimization techniques.

The paper is organized as follows: Section 2 describes the problem and challenges associated with it. Sections 3 and

4 describe our proposed models and optimization schemes. Finally, Sections 5 and 6 discuss experimental results.

2. PERSONAL QUESTION ANSWERING

The problem is framed as a retrieval-based QA system over stored memories. More formally, given a question q and a set of user memories M_q, the system returns a subset of memories R_q which are relevant and answers the spoken question. Throughout the paper, the set $\{q, M_q, R_q\}$ is referred to as a QA group. Table 1 shows an example QA group of four memories marked as relevant or irrelevant to the input question.

For solving this problem, a classification approach could be adopted given question and memory pairs but this approach is not optimal due to the large class imbalance between relevant and irrelevant user memories for each question as shown in Table 4. A better end-to-end formulation should take into account all user stored utterances (memories) when making relevance decisions for each individual memory, i.e. directly optimizing for $F1$ score for each QA group. However, it is challenging to directly optimize for $F1$ measure due to its discrete, non-differentiable nature.

Another challenge rises from the spoken nature of the presented system, where both user memories and questions are transcribed by an ASR system. Due to varying acoustic conditions, names or locations could be recognized in two different ways during storing a memory and recalling it, which are naturally many days or weeks apart. We found that this effect compounds the effect of ASR errors on the end-to-end retrieval performance.

3. NEURAL RETRIEVAL MODELS

We propose various optimization objectives and model architectures for determining the relevance of stored memories given a question. None of our proposed architectures contain any recurrent units because fast and efficient inference is key to a seamless user experience. We focus our efforts on optimization in this work, and demonstrate that effects of carefully constructed optimization objectives, to train a relatively simple model architecture to achieve high performance.

The input to a model is a question, q, and its corresponding set of memories, $M_q = \{m_1, ..., m_{|M_q|}\}$. Each question and stored memory undergoes a string preprocessing step to clean up the text and tokenize the utterance into words. The utterance is then encoded using word-level representations, for input to the model. We also experiment with using compositional word embeddings [21, 22] to distill task-specific subword knowledge into our model. Specifically, for a given query $q = (w_1, ..., w_{|q|})$, we define its feature vector to be the matrix of word embeddings:

$$E_q = [\mathbf{v}_{w_1}, ..., \mathbf{v}_{w_{|q|}}] \in \mathbb{R}^{|q| \times d}.$$

where $|q|$ denotes the length of the question in words and d is the feature dimension. We use similar terminology for the vector representing a memory, which we write here as $E_{m_i} \in \mathbb{R}^{|m_i| \times d}$, where $|m_i|$ denotes the number of tokens in the memory. We also experiment with modifying the word embedding to include compositional word embeddings, as a way of leveraging the task-specific information present in our corpus. The compositional word embedding module generates word representations using character embeddings, following closely the architecture of the `CharCNN` model presented in [22]. This module is jointly trained with the rest of the model using the task-specific objective. Each word can then be represented by the concatenation of its pre-trained word vector and the `CharCNN` embedding.

In the next sections, we explain the architectures of the models `TEFF` and `TEFFCH`[1].

3.1. `TEFF` and `TEFFCH`

The `TEFF` model is comprised of N fully-connected layers, followed by a max-pool layer across time to produce a fixed-dimensional vector. The query and memory embeddings are both processed through the same network, producing h-dimensional vectors for the query and memory, represented by u and v, respectively. The joint activation and logits are computed as:

$$uv = \mathtt{concat}(u, v, |u - v|, u \odot v)$$
$$\mathtt{logits} = \mathtt{softmax}(\mathtt{dropout}(uv)) \in \mathbb{R}^2$$

The `TEFFCH` model follows a similar structure, except the input embedding is given by the concatenation of the pre-trained word vector and the `CharCNN` embedding, that is jointly trained, producing an end-to-end model.

4. DIRECT OPTIMIZATION OF F-MEASURE

As the goal of our model is to correctly assign the label 'relevant' vs. 'irrelevant' to a memory given a question, we can formulate the optimization objective as the maximization of labelling accuracy. Expressed as a loss function, we try to minimize the cross-entropy between the estimated class probabilities and the ground truth label distribution for a set of question-memory pairs:

$$\mathcal{L}_{cs}(\theta) = -\sum_i \sum_j^{|M_q|} \sum_c \log p_\theta(m_j|q_i) \mathbb{I}_c(y_{ij}) \qquad (1)$$

m_j and q_i denote the memory and question pair respectively, y_{ij} is corresponding label, $\mathbb{I}$ is the indicator function, and p_θ denotes the model, parametrized by θ. Even though this formulation renders optimization straightforward, it leads to a

[1] Model name(s) used for simplicity.

discrepancy at evaluation time as we optimize for one objective during training but use another metric for model evaluation. More specifically, we optimize for maximum accuracy of question-memory-pair labelling during training but evaluate our model using the $F1$ score averaged across all QA groups. An analogous objective cannot be used as an optimization for $F1$, as $F1$ is not differentiable. Furthermore, Eq. 1 does not address the large class imbalance between irrelevant memories versus relevant memories in a QA group.

To address this discrepancy, we propose a novel optimization objective that can directly estimate the evaluation metric.

4.1. Policy Gradient based Approximation

Our goal is to maximize the expected $F1$ score, for a given dataset. To do this, we first formulate our task as a reinforcement learning problem in which our network acts as the agent, i.e. policy network, and so provides the probability for taking a particular action on each question-memory pair:

$$\mathcal{L}_{\text{re}}(\theta) = -\mathbb{E}_{a^i \sim p_\theta(M_q|q)}[R(a^{M_q})] \tag{2}$$

where p_θ is the policy network, $R(a^{M_q})$ is the reward function, and a^i is the action given (m_i, q). Since the reward is not differentiable, we use REINFORCE [18, 23] to estimate the gradients. Based on this algorithm, the gradients are calculated as follows:

$$\nabla_\theta \mathcal{L}_{\text{re}}(\theta) = -\mathbb{E}_{a_i \sim p_\theta}[R(a^{M_q}) \cdot \nabla_\theta \log p_\theta(M_q|q)] \tag{3}$$

Even though this new formulation has the potential to boost model performance, it presents several challenges. Firstly, the score can only be calculated for an entire QA group, i.e. a query and all the associated memories. To address this, we modify the batching strategy so that each batch contains one query and all the associated memories. Secondly, Eq. 2 algorithm is hard to optimize especially if the optimization starts from scratch. To resolve this problem, we use curriculum learning [24] under a multi-task learning (MTL) framework. We firstly train the model using Eq. 1 to kick-start training. We then start training the model using the following MTL at a reduced learning rate:

$$\mathcal{L}(\theta) = (1-\lambda) \cdot \mathcal{L}_{cs}(\theta) + \lambda \cdot \mathcal{L}_{re}(\theta) \tag{4}$$

The new batching strategy whereby a batch consists of an entire QA group is used when the optimization function is set to Eq. 4. λ is a hyper-parameter and is determine using random search on validation set. The reward function is explained in more detail in Sec. 4.1.1.

In addition to aforementioned challenges, Eq. 2 and in general REINFORCE [18] suffers from high variance given its inherent nature of noisy gradient estimates. Selecting the right reward function plays an important role to reduce the variance of gradient estimator [19]. Motivated by this observation and previous works [19, 20], we use the exact score at test time to *baseline* Eq. 2:

$$\nabla_\theta \mathcal{L}_{\text{re}}(\theta) = (R(a^{M_q}) - \mathtt{F1}(M_q)) \cdot \nabla_\theta \log p_\theta(M_q|q) \tag{5}$$

where F1 is the exact scoring function that is used during test time. We sample an action according to

$$\hat{a}_i = \begin{cases} 1 & \text{if } c_i \text{ is relevant } \& \ p_{c_i} \geq \zeta \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

where

$$c_i = \text{argmax}_{a_j} p_\theta(a_j|m_i, q)$$

is the greedy output of the model, and $\hat{a}$, i.e. $\hat{a} = \{\hat{a}\}_{i=1}^{|M_q|}$, is the set of predications for a question and its memories and ζ is the confidence threshold of the predictions. Using these predictions, $\mathtt{F1}(M_q)$ can be easily calculated for a question and memory group. This method not only helps to reduce the variance by *baselining* Eq. 2 but also helps the model to make predications with high confidence, given ζ. The ability to directly specify the predication confidence as part of the objective is a distinct advantage over previous methods [16].

4.1.1. Reward Function design

Designing an effective reward strategy for use in Eq. 2 is critical for successful training. This also applies to reinforcement learning in general. Using the vanilla $F1$ score as the reward can have several side-effects. For example, if all the predictions of the model are incorrect, then the $F1$ score becomes zero and, as a result, the loss function become zero and no errors are backpropagted to the network. To address these issues, we define a modified reward function as follows:

$$\text{reward} = \begin{cases} 1.0 & \text{if A } \& \ \forall \text{P} \\ -0.1 & \text{if A } \& \ \forall \text{R} \\ \textit{accuracy} & \text{if A } \& \ \exists \text{P} \\ -0.5 & \text{if } tp == 0 \\ -0.01 & \text{if } 0 \leq \mathtt{F1} \leq 0.2 \\ \mathtt{F1} & \text{otherwise} \end{cases}$$

where A means there are no ‘relevant’ ground truth labels in the QA group, P means hypothesized ‘irrelevant’ label is correct, R means hypothesized ‘irrelevant’ label is incorrect, *tp* denotes number of true positives in QA group, and *accuracy* is the classification accuracy.

5. EXPERIMENTS

5.1. Datasets

Our data consists of a total approximately 20,000 QA groups divided into the datasets ‘train’, ‘dev’, ‘TEST-1’ and ‘TEST-2’. Each QA group contains one question. For ‘TEST-2’, the question is an utterance chosen at random from experimental personal assistant logs in which the user has asked the assistant to retrieve a personal memory. For 'TEST-1', the user question was typed in by an annotator to resemble an actual user utterance. Table 2 shows the approximate number of QA groups per dataset in thousands (K).

dataset	number of QA groups	number of answers
train	~14K	~308K
dev	~1K	~61K
TEST-1	~8K	~105K
TEST-2	~3K	~151K
all	~26K	~626K

Table 2. Approximate number of QA groups per dataset

The answers in each QA group consist of anywhere between 1 to 81 memories which a user had asked the assistant to remember. With the exception of the manually entered questions in 'TEST-1', the questions and memories are the output of the assistant's speech recognition engine and, as a result, contain speech recognition errors. Each of the memories has been manually annotated as being relevant or irrelevant to the question in its QA group.

As is apparent in Table 3, the number of memories per QA group differs significantly across datasets. This is because the number of memories each user has stored varies greatly and the maximum number of memories annotated per QA group varied between annotation groups.

dataset	min	max	mean	std. dev.
train	1.0	80.0	21.34	19.35
dev	18.0	81.0	49.26	29.91
TEST-1	1.0	30.0	13.76	10.45
TEST-2	18.0	81.0	49.70	29.91
all	1.0	81.0	23.70	22.83

Table 3. Minimum, maximum and mean number of memories across QA groups in each dataset.

There is a significant class imbalance between relevant and irrelevant memories across datasets as most of a user's memories are not relevant to a given question. Table 4 shows the percent of memories with a 'relevant'-label in each QA group averaged over all the QA groups in dataset. Because of this imbalance, relevant answers were upsampled using weighted sampling during training to create batches with roughly the same number of relevant and irrelevant examples.

Each question and answer also undergoes a preprocessing step in which contractions and abbreviations are expanded, e.g. "doesn't" → "does not", "wanna" → "want to", common question carrier phrases are removed, e.g. "can you remember what", "please tell me who" and stop words with little meaning are deleted, e.g. "did", "does", "is". Such preprocessing makes the wording more consistent across utterances and removes words and phrases with little or no semantic content. It decreases the average number of tokens in questions from 6.6 to 3.8 and in answers from 4.2 to 3.7. The change in number of tokens for each data is listed in Table 5.

dataset	% relevant memories
train	15.08%
dev	4.54%
TEST-1	20.94%
TEST-2	4.59%
all	15.08%

Table 4. Percentage of memories with 'relevant'-label in each QA group averaged over all QA groups in dataset

	raw		preprocessed	
dataset	question	answer	question	answer
train	6.8	3.7	4.0	3.3
dev	7.0	3.8	4.1	3.4
TEST-1	6.0	6.7	3.3	5.6
TEST-2	7.1	3.8	4.2	3.4
all	6.6	4.2	3.8	3.7

Table 5. Average number of tokens per question and per answer in each dataset

5.2. Evaluation metrics

Each model was evaluated on the test sets 'TEST-1', where the questions were typed in by annotators, and 'TEST-2', where questions are the output of an ASR engine. For each QA group in the dataset, the precision, recall and $F1$ score were calculated by comparing the relevance labels assigned by annotators with the hypotheses returned by the model. The precision, recall and $F1$ score of all QA groups in each dataset were then averaged to give the average precision, average recall and average $F1$ score for the entire dataset. Ranking of memories was not considered.

5.3. Model specifications

We report results on the `TEFF` and `TEFFCH`. We used the Adam optimizer [25] to train all models with Eq. 1 and use a constant learning rate of 0.001. When the training objective is switched to Eq. 4, we adopt a different batching scheme and decay the learning rate by a factor of 0.1. We use the `ReLu` activation function throughout our models. We use L2 weight decay to train our models and apply dropout at a rate of 0.1 across all models. We use a batch size of 128 and pick the model with the best performance on the validation set. We use random search for hyperamater tuning to determine the best model configuration. We use two variations of batching for our experiments. For models trained using Eq. 1, we construct a batch of query-memory pairs using random sampling, but oversample from the positive examples to ensure a $1:1$ ratio of positive and negative examples in every batch. However, when using the MTL objective in Eq. 4, we batch all memories for a given question to compute the $F1$ score for the given QA group. For models that only consume word embeddings, we batch the queries and memories to a maximum sentence length, and all sentences smaller than this length were padded using a <PAD> token. For all our models, we use 300 dimensional pre-trained `fastText` word vectors [26], trained on Wikipedia data from 2017, news datasets from `statmt.org` from 2007-2016 as well as the UMBC corpus [27]. We found that a maximum utterance length of 10 was sufficient to ensure good results, given that the average query length was much shorter (after preprocessing). For models with a `CharCNN` module, the character level inputs were also padded at the word-level using a maximum word length of 8.

Our best `TEFF` model is 2-layer network with 694 units each. The best `TEFFCH` model employs two convolution layers of with a kernel size of 1 and 2, and 128 filters each, followed by a linear layer outputting 108 dimensional `CharCNN` embeddings, concatenated together with pre-trained word vectors to give 408 dimensional word representations. The concatenated embeddings are then processed through a 2-layer network with 736 units each. All models have a final softamx layer to output a distribution over two classes.

5.4. Smooth Approximation

In order to have a comprehensive evaluation of our proposed method, we compare our method with [16], wherein a smooth approximation of the $F1$ score was proposed. This smooth objective is differentiable and is formulated as follows:

$$\mathcal{L}_{fs}(\theta) = -\sum_j \mathcal{F}_\theta(M_{q_j}|q_j) \tag{7}$$

$$\text{PR}_\theta = \frac{\sum_i \log p_\theta \mathbb{I}_{tp}(q_j, m_i)}{\sum_i \log p_\theta \mathbb{I}_{tp}(q_j, m_i) + \sum_i \log p_\theta \mathbb{I}_{fp}(q_j, m_i)} \tag{8}$$

$$\text{RE}_\theta = \frac{\sum_i \log p_\theta \mathbb{I}_{tp}(q_j, m_i)}{\sum_i \log p_\theta \mathbb{I}_{tp}(q_j, m_i) + \sum_i \log p_\theta \mathbb{I}_{fn}(q_j, m_i)} \tag{9}$$

$$\mathcal{F}_\theta = 2 \cdot \frac{\text{PR}_\theta \cdot \text{RE}_\theta}{\text{PR}_\theta + \text{RE}_\theta} \tag{10}$$

where $\mathbb{I}_{tp}$ is the indicator function for true positives, $\mathbb{I}_{fp}$ for false positives, and $\mathbb{I}_{fn}$ for false negatives. Since $\mathcal{F}$ is a differentiable function and is parameterized by θ, we can directly optimize for it during training. We followed the same steps as for the MTL loss (Eq. 4), i.e. the different batching strategy and decayed learning rate.

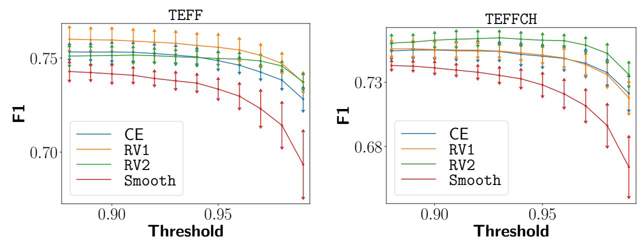

Fig. 1. Plot comparing 'TEST-1' performance for the various aforementioned training objectives. The error bars were generated by running multiple trials of the same model, with different seeds.

Model	Objective	$t = 0.97$	$t = 0.98$	$t = 0.99$	# Params
TEFF	RV1	2.283%	2.295%	2.259%	0.70M
	RV2	1.075%	1.772%	2.766%	
	Smooth	-1.673%	-1.832%	-2.405%	
TEFFCH	RV1	1.807 %	2.896 %	5.016%	0.89M
	RV2	3.714%	4.732%	5.239%	
	Smooth	-1.557%	-1.413%	-1.407%	

Table 6. Table showing the relative improvement in $F1$ score achieved using various training objectives across different thresholds, compared to the cross-entropy baseline, on 'TEST-1'. The model size is given by **# Params**.

One of the main advantages of our proposed method compared to the smooth formulation, is that we can directly enforce the confidence level in the prediction as in Eq. 6. Moreover, the results show that our proposed method significantly outperforms the above smooth formulation.

6. RESULTS AND ANALYSIS

A key aspect of our proposed objective is that the model is made aware of the type of errors[2] in its prediction, and can use this information to trade-off the number of false positives and false negatives, for an optimal $F1$ score. In our experiments, when training with Eq. 1, the model is able predict, with reasonable accuracy, the relevant memories, for various query types. However, for some QA groups that contain noise in the form of ASR errors or otherwise complex queries, this accuracy is lower as expected. For these challenging cases, our method encourages the network to balance the number of false positives and false negatives, to avoid predictions with high precision and low recall (and vice-versa), but rather maintain an optimal balance of the two.

We investigate the gains realized when optimizing using our MTL objective in Eq. 4 and compare it to using the standard loss formulation (Eq. 1) and smooth loss (Sec. 5.4) as a training objective. As shown in Table 6 and Figure 1, we find that our proposed optimization objective, infact leads to

[2] Type I - False positive or Type II - False negative

an improvement in $F1$ score for this QA task. In Figure 1, the error bars were generated by setting different seeds for each trial. However, we fix the curriculum learning strategy beforehand, and do not tune it for each trial. This may be sub-optimal for the performance of the model. Tuning this curriculum learning strategy per trial will likely show futher gains. Figures 2 and 3 show the performance of various models trained with different objectives, across our test sets. In our results, `RV1` and `RV2` refer to training using Eqs. 3 and 5 respectively, and `Smooth` refers to the objective described in Section 5.4.

We observe that the underlying structure of 'TEST-1' and 'TEST-2' are different, and hence show different gains in performance. Both test sets and the training set have on average, about 3 relevant memories per QA group, but 'TEST-2' has more total memories per QA group, and shorter memories, on average (Table 5). We find that 'TEST-2' shows smaller gains relative to 'TEST-1', when using our objective, as 'TEST-2' contains shorter utterances, which provide less context to distinguish different memories. However, on 'TEST-1', our objective was able to learn a stronger semantic model which is able to adapt to correctly identify paraphrasing, ASR errors and long-range context in complex utterances (Figure 3).

Moreover, we expected larger gains using `CharCNN`, but this was not always the case. We hypothesize that this is due to the short nature of utterances in our *training* data, which are only 4 tokens long, on average (see Table 5), and can be suitably encoded using pre-trained word vectors. The pre-trained word vectors are trained on a much larger corpus and generalize well as they aggregate contextual information from multiple domains. On the other hand, our `CharCNN` module is not processed through a sequential network, and hence lacks inter-word context. This module is also trained on our task-specific loss which may not be optimal for learning such embeddings, and can depict quite pathological behviour, given the size of the dataset and the length of utterances. This can cause the model to produce overloaded representations and make it prone to overfitting, thereby harder to train. We hypothesize that with less pre-processing, the additional context available could mitigate some of the issues exhibited by the `CharCNN` module, leading to futher gains.

7. CONCLUSION

In this paper, we present an end-to-end system for spoken personal question answering. Moreover, we propose a novel objective function to directly optimize the $F1$ measure for our information retrieval task. By directly optimizing the $F1$-score, we can take into account the predicted labels of *all* answers simultaneously. It also enables us to take the types of errors into consideration during optimization, e.g. number of false positives, number of false negatives. Furthermore, our proposed objective can mitigate the effects of class imbalance and noisy data in the form of ASR errors. Our extensive experimentation shows that the aforementioned approaches deliver benefits to system performance. We also analyze the impact of our methods on different datasets with varying structure.

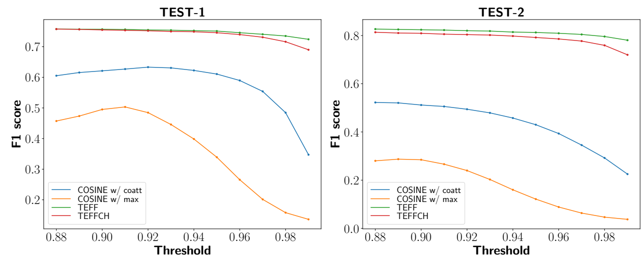

Fig. 2. Performance of various models across our test sets. The `COSINE` models is a parameter-free Bag-of-Words (BOW) model that consumes word embeddings, which are aggregated using 1) `COATT`, which uses a coattention encoding similar to the one presented in [6] and 2) `MAX`, which simply does a max-pooling across time to produce fixed dimensional vectors. These words are computed for both the query and memories, and a memory is deemed relevant if the cosine similarity between the query and memory vectors exceeds a certain threshold.

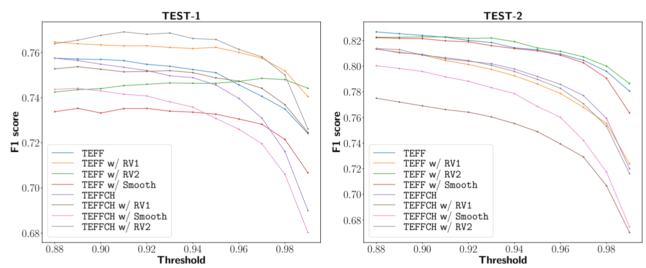

Fig. 3. Plot showing the evolution of $F1$ scores across various thresholds, and across models `TEFF` and `TEFFCH`. Both models show an improvement when optimizing using Eq. 4, while the `Smooth` $F1$ objective as described in section 5.4 performs worse than our baseline model (Eq. 1). `RV1` refers to models trained using Eq. 3, whereas `RV2` refers to the reduced variance *baseline* described in Eq. 5.

8. REFERENCES

[1] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al., "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[2] G. E. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 1, pp. 30–42, Jan 2012.

[3] Luke S Zettlemoyer and Michael Collins, "Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars," *arXiv preprint arXiv:1207.1420*, 2012.

[4] R. De Mori, F. Bechet, D. Hakkani-Tur, M. McTear, G. Riccardi, and G. Tur, "Spoken language understanding," *IEEE Signal Processing Magazine*, vol. 25, no. 3, pp. 50–58, May 2008.

[5] Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov, "Towards ai-complete question answering: A set of prerequisite toy tasks," *arXiv preprint arXiv:1502.05698*, 2015.

[6] Caiming Xiong, Victor Zhong, and Richard Socher, "Dynamic coattention networks for question answering," *ICLR*, 2017.

[7] Alessandro Sordoni, Phillip Bachman, and Yoshua Bengio, "Iterative alternating neural attention for machine reading," *CoRR*, vol. abs/1606.02245, 2016.

[8] Steve Young, Milica Gašić, Simon Keizer, François Mairesse, Jost Schatzmann, Blaise Thomson, and Kai Yu, "The hidden information state model: A practical framework for pomdp-based spoken dialogue management," *Computer Speech & Language*, vol. 24, no. 2, pp. 150–174, 2010.

[9] Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao, "Deep reinforcement learning for dialogue generation.," in *EMNLP*, 2016, pp. 1192–1202.

[10] Ruhi Sarikaya, "The technology behind personal digital assistants: An overview of the system architecture and key components," *IEEE Signal Processing Magazine*, vol. 34, no. 1, pp. 67–81, Jan 2017.

[11] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang, "Squad: 100,000+ questions for machine comprehension of text," *arXiv preprint arXiv:1606.05250*, 2016.

[12] Bhaskar Mitra and Nick Craswell, "Neural models for information retrieval," *arXiv preprint arXiv:1705.01509*, 2017.

[13] Zhiguo Wang, Wael Hamza, and Radu Florian, "Bilateral multi-perspective matching for natural language sentences," *arXiv preprint arXiv:1702.03814*, 2017.

[14] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi, "Bidirectional attention flow for machine comprehension," *arXiv preprint arXiv:1611.01603*, 2016.

[15] Quoc V. Le and Alexander J. Smola, "Direct optimization of ranking measures," *CoRR*, vol. abs/0704.3359, 2007.

[16] Joan Pastor-Pellicer, Francisco Zamora-Martínez, Salvador España Boquera, and María José Castro-Bleda, "F-measure as the error function to train neural networks," in *Proceedings of the 12th International Conference on Artificial Neural Networks: Advances in Computational Intelligence - Volume Part I*. 2013, IWANN'13, pp. 376–384, Springer-Verlag.

[17] Wenduan Xu, Michael Auli, and Stephen Clark, "Expected f-measure training for shift-reduce parsing with recurrent neural networks," in *HLT-NAACL*, 2016.

[18] Ronald J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Mach. Learn.*, vol. 8, no. 3-4, pp. 229–256, May 1992.

[19] Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba, "Sequence level training with recurrent neural networks," *ICLR*, vol. abs/1511.06732, 2016.

[20] Steven J. Rennie, Etienne Marcheret, Youssef Mroueh, Jarret Ross, and Vaibhava Goel, "Self-critical sequence training for image captioning," in *CVPR*, pp. 1179–1195. 2017.

[21] Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush, "Character-aware neural language models.," in *AAAI*, 2016, pp. 2741–2749.

[22] Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu, "Exploring the limits of language modeling," *arXiv preprint arXiv:1602.02410*, 2016.

[23] Wojciech Zaremba and Ilya Sutskever, "Reinforcement learning neural turing machines," *CoRR*, vol. abs/1505.00521, 2015.

[24] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston, "Curriculum learning," in *ICML*, New York, NY, USA, 2009, ICML '09, pp. 41–48, ACM.

[25] D Kinga and J Ba Adam, "A method for stochastic optimization," in *International Conference on Learning Representations (ICLR)*, 2015, vol. 5.

[26] Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin, "Advances in pre-training distributed word representations," in *LREC*, 2018.

[27] Lushan Han, Abhay L Kashyap, Tim Finin, James Mayfield, and Jonathan Weese, "Umbc_ebiquity-core: semantic textual similarity systems," in *Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, 2013, vol. 1, pp. 44–52.

[28] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin, "Attention is all you need," in *NIPS*, pp. 5998–6008. 2017.

[29] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber, "Highway networks," *CoRR*, vol. abs/1505.00387, 2015.

[30] Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber, "Training very deep networks," in *NIPS*, pp. 2377–2385. 2015.

[31] Tao Qin, Tie-Yan Liu, and Hang Li, "A general approximation framework for direct optimization of information retrieval measures," *Information Retrieval*, vol. 13, no. 4, pp. 375–397, Aug 2010.

[32] Krzysztof J. Dembczynski, Willem Waegeman, Weiwei Cheng, and Eyke Hüllermeier, "An exact algorithm for f-measure maximization," in *NIPS*, pp. 1404–1412. 2011.

[33] Susan Dumais, Edward Cutrell, JJ Cadiz, Gavin Jancke, Raman Sarin, and Daniel C. Robbins, "Stuff i've seen: A system for personal information retrieval and re-use," in *SIGIR*. 2003, pp. 72–79, ACM.

[34] Chris J.C. Burges, "From ranknet to lambdarank to lambdamart: An overview," Tech. Rep., June 2010.

[35] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang, "Semantic parsing on freebase from question-answer pairs," in *EMNLP*, 2013, pp. 1533–1544.

[36] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008.

[37] Daniel (Zhaohan) Guo, Gokhan Tur, Scott Wen-tau Yih, and Geoffrey Zweig, "Joint semantic utterance classification and slot filling with recursive neural networks," in *IEEE SLT*, December 2014.