A REINFORCEMENT LEARNING-DRIVEN TRANSLATION MODEL FOR SEARCH-ORIENTED CONVERSATIONAL SYSTEMS

Wafa Aissa, Laure Soulier, and Ludovic Denoyer
Sorbonne Université, CNRS, LIP6,F-75005 Paris, France
prenom.nom@lip6.fr

ABSTRACT

In this research, we aim to apply deep learning models in order to empower search oriented conversational systems (SOCS). While using chat-bots or personal assistants, users tend to express their information needs in a conversational form using natural language queries However, natural language queries degrade the retrieval performance of search engines which are optimized for keyword queries. We focus here on the understanding of NL expressions for building keyword based queries. We propose a reinforcement learning-driven translation model framework able to 1) learn the translation from NL expressions to queries in a supervised way, and, 2) to overcome the lack of large-scale dataset by framing the translation model as a word selection approach and injecting relevance feedback in the learning process. Experiments are carried out on two TREC datasets and outline the effectiveness of our approach. This work is a continuation of our paper Aissa et al. (2018).

1 INTRODUCTION

Artificial Intelligence, and more particularly deep learning, have recently opened tremendous perspectives for reasoning over semantics in text-based applications such as machine translation (Lample et al., 2017), chat-bot (Bordes & Weston, 2016), knowledge base completion (Lin et al., 2015) or extraction (Hoffmann et al., 2011). Very recently, conversational information retrieval (IR) has emerged as a new paradigm in IR (Burtsev et al., 2017; Joho et al., 2018), in which natural conversations between humans and computers are used to satisfy an information need. As for now, conversational systems are limited to simple conversational interactions (namely, chit-chat conversations) (Li et al., 2016a; Ritter et al., 2011), closed worlds driven by domain-adapted or slot-filling patterns (Bordes & Weston, 2016; Wang & Lemon, 2013) (e.g., a travel planning task requiring to book a flight, then a hotel, etc...) or knowledge-base extraction (e.g., information extraction tasks) (Dhingra et al., 2017).

In contrast, search-oriented conversational systems (SOCS) aim at finding information in an open world (both unstructured information sources and knowledge-bases) in response to users' information needs expressed in natural language (NL); the latter often being ambiguous. Therefore, one key challenge of SOCS is to understand users' information needs expressed in NL to identify relevant documents.

Formulating an information need through queries has been outlined as a difficult task (Vakulenko et al., 2017; Agichtein et al., 2006; Joachims, 2002) which is generally tackled by refining/reformulating queries using pseudo-relevance feedback or users' clicks. In SOCS, there is an upstream challenge dealing with the building of the query from a NL expression that initiates the search session to avoid useless users' interactions with the system. This problem could be tackled for instance through deep neural translation models (e.g., encoder-decoder approaches)as initiated by Song et al. (2017); Yin et al. (2017). However, these methods learn the query formulation model independently of the search task at hand. To overpass this limitation, Nogueira & Cho (2017) have proposed a reinforcement learning model for query reformulation in which the reward is based on terms of documents retrieved by the IR system.

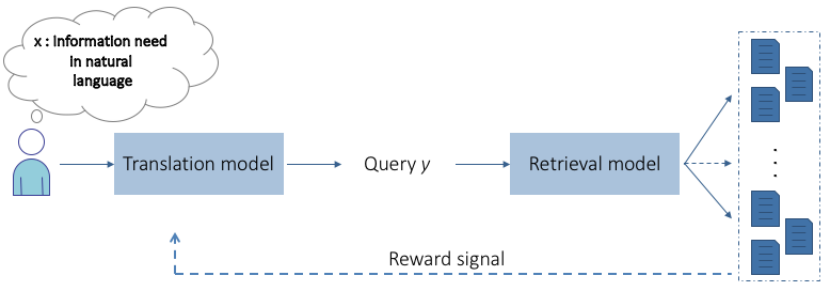

Figure 1: Overview of our reinforcement learning-driven translation model for SOCS

In this work, we propose to bridge these two lines of work: 1) machine translation to learn the mapping between information needs expressed in NL and information needs formulated using keywords (Song et al., 2017; Yin et al., 2017), and 2) reinforcement learning to inject the task objectives within the machine translation model (Nogueira & Cho, 2017). More particularly, we propose a two-step model which first learns the translation model through the supervision of NL-query pairs and then refines the translation model using a relevance feedback provided by the search engine. We define different variants of our model based on whether or not to consider a first continuous transformation of the overall information need before the translation and design a memory than keeps the continuous transformation of the chosen words by our model.

It is worth mentioning that it does not exist SOCS-oriented dataset that both aligns users' information needs in NL with keyword-based queries and includes a document collection to perform a retrieval task. To the best of our knowledge, TREC datasets are the only ones expressing such constraint, but the number of NL-query pairs is however limited. To fit with the issue of dealing with large vocabulary and the dataset constraint, we frame the translation model as a word selection one which aims at identifying which words in the NL expression can be used to build the query. Our model is evaluated on two TREC datasets. The obtained results outline the effectiveness of combining reinforcement learning with machine translation models.

The remaining of the paper is organized as follows. Section 2 details our translation model. Section 3 presents the evaluation protocol and results are highlighted in Section 4. The conclusion and perspectives are discussed in Section 5.

2 Reinforcement learning-driven translation model

2.1 Notation and problem formulation

Our reinforcement learning-driven translation model allows to formulate a user's information need x expressed in NL into a keyword-based query y. The user's information need x is a sequence of n words ($x = x_1, ..., x_i, ..., x_n$). To fit with our word selection objective, the query y is modeled as a binary vector $y \in \{0, 1\}^n$ of size n (namely, the size of the natural language expression x). Each element $y_j \in y$ equals to 1 if word $x_i \in x$ exists in query y and 0 otherwise. For example, if we consider the NL as "Identify documents that discuss sick building syndrome or building related illnesses." and the key-words query as "sick building syndrome.", the expected query will be formulated as follows: $y = (0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0)$.

The objective of our model f_θ (with θ being the parameters of our model) is to estimate the probability $p(y|x)$ of generating the binary vector y given the NL expression x. Since terms are not independent within the formulation of NL expressions and queries, it makes sense to consider that the selection of a word is conditioned by the sequence of decisions taken on previous words $y_{<i}$. Thus, $P(y|x)$ could be written as follows:

$$p(y|x) = \prod_{y_i \in y} p(y_i|y_{<i}, x)$$

This probability is first learned using a maximum likelihood estimation (MLE) on the basis of NL-query pairs (Section 2.2). Then, this probability is refined using reinforcement learning techniques (Section 2.3). We end up with the network architecture used in the translation model.

2.2 SUPERVISED TRANSLATION MODEL: FROM NL TO QUERIES

The translation model works as a supervised word selection model aiming at building queries y by using the vocabulary available in NL expressions x. To do so, we use a set D of N NL-query pairs $D = \{(x^1, y^1), ..., (x^k, y^k), ..., (x^N, y^N)\}$. The objective of the translation model is to predict whether each word x_i^k in the NL expression x^k is included in the expected query y^k. In other words, it consists in predicting the probability $p(\hat{y}_i^k = y_i^k | \hat{y}_{<i}^k, x^k)$ that the i^{th} element $\hat{y}_i^k$ of vector $\hat{y}^k$ is equal to the same element y_i^k in the original query y^k (namely, that $\hat{y}_i^k = y_i^k$) given the state of previous elements $\hat{y}_{<i}^k$ and the NL expression x^k. This probability $p(\hat{y}_i^k = y_i^k | \hat{y}_{<i}^k, x^k)$ is modeled using a Bernoulli distribution in which parameters are estimated through the probability distribution. Let's define for a NL-query instance (x^k, y^k), $f_{(\theta, x^k)} = \sum_{y_i^k \in y^k} log(p(\hat{y}_i^k = y_i^k | \hat{y}_{<i}^k, x^k))$. The translation model is trained by maximizing the following MLE over the set D of NL-query pairs (x_k, y_k):

$$L_{SMT} = \sum_{(x^k, y^k) \in D} log(f(\theta, x^k))$$

2.3 REINFORCEMENT LEARNING

To inject the task objective in the translation model, we consider that the process of query building could be enhanced through reinforcement learning techniques. Therefore, the word selection could be seen as a sequence of choices of selecting word x_t at each time step t. The choices are *rewarded* at the end of the selection process by a metric measuring the effectiveness of the query building process within a retrieval task. Particularly, the predicted query $\hat{y}$ obtained from the binary vector $\hat{y}$ is fed to a retrieval model to rank documents. For each NL expression x (and accordingly the associated predicted query $\hat{y}$), we dispose of a set $\mathcal{D}_x$ of relevant documents (also called ground truth). We note GT the set of n pairs $(x; \mathcal{D}_x)$. With this in mind, the effectiveness of the obtained ranking could be estimated using an effectiveness-driven metric (e.g., the MAP). Thus, the reward R for a generated query $\hat{y}$ given the relevance feedback pair $(x, \mathcal{D}_x)$ is obtained as follows:

$$R(\hat{y}) = MAP(\hat{y}, \mathcal{D}_x)$$

At the end of the selection process, the objective function aims at maximizing the expectation of the search effectiveness over the predicted queries:

$$L_{RL}(\theta) = arg\max_{\theta} \; \mathrm{E}_{\substack{(x; \mathcal{D}_x) \in GT \\ \hat{y} \sim f_\theta(x)}} [R(\hat{y}) - \bar{R}]$$

where $\hat{y}$ is given by the translation model $f_\theta(x)$ and $\bar{R}$ equals to the average reward of the past epoch, this will push the model to yield better outcomes than the previous ones during training phase. This objective function is maximized using gradient descent techniques from Baxter et al. (1999).

2.4 MODEL ARCHITECTURE

To tackle our selection problem using recurrent neural networks, we recall that the aim of our model is to select the relevant words from the NL information need x. Our decoder has a binary output function that at each time step t takes the current word x_i from the original natural language query x and predicts the probability of the word's effectiveness for the information need (0 for selecting a word ans 1 for discarding it). We identify different variants of our model:

1. We make the assumption that capturing the overall information need before decoding it may help the model to select the relevant words based on that. This can be done using an encoder to transform the NL sentence to its continuous representation and then decoding it. This hypothesis engender two different versions of our model:
   (a) Discard this hypothesis and directly pass x to the decoder, the first hidden state of the decoder is initialized by a zero vector.

(b) Pass x through an encoder and then initiate the first hidden state of the decoder with the last hidden state of the encoder $h_n = d_0$.

(c) Use $h_n = d_0$ and inject h_n with every input word when decoding the NL sentence.

2. Since we framed our problem as selecting the most important words from x and we are using RNNs to decode the sentence, we believe that creating a memory that keeps a combined representation of the already selected words may be helpful, we identify two different ways to do this:

   (a) Discard this idea to analyze the models outcomes without a memory decoder.

   (b) Use two different decoding functions i.e. a binary memory, the first one is applied when the previous word was selected $y_{i-1} = 1$ and the second one is applied when the previous word is not selected $y_{i-1} = 0$.

   (c) Another idea is to use two hierarchical decoders, the first LSTM decodes the overall information need and at each time step t takes the current word x_i from the original natural language query x and predicts the probability of the word's effectiveness for the information need while the second LSTM serves as a memory of the overall information need x and the already selected words $y_{<i}$ s.t $y_i = 1$. So intuitively we only pass the words selected by the first decoder through the second decoder.

We summarize in 1 the different variants discussed above and check the implemented ones:

	1.(a)	**1.(b)**	**1.(c)**
2.(a)		SMT+RL 1	
2.(b)	SMT+RL 2	SMT+RL 3	SMT+RL 5
2.(c)		SMT+RL 4	

Table 1: Our 5 proposed models based on their implementation schemes

Overall, the model is based on a RNN to build a query $\hat{y}$ from the input x. Particularly, each element x_i of x is modeled through word embeddings w_{x_i}; resulting in a sequence w_x of word embeddings for input x.

In this work we only try 5 combinations of the variants discussed above and keep the other combinations for future work, this results in:

- **SMT+RL 1:** This is the basic version, where we don't keep track of the already selected words and just use the last hidden state of the encoder h_n as the first hidden state of the decoder. And at each time step the hidden state of the decoder d_t is updated by:

$$d_0 = h_n$$

$$d_t = f(d_{t-1}, x_t)$$

With f a LSTM layer and x_t is the current word to select or discard.

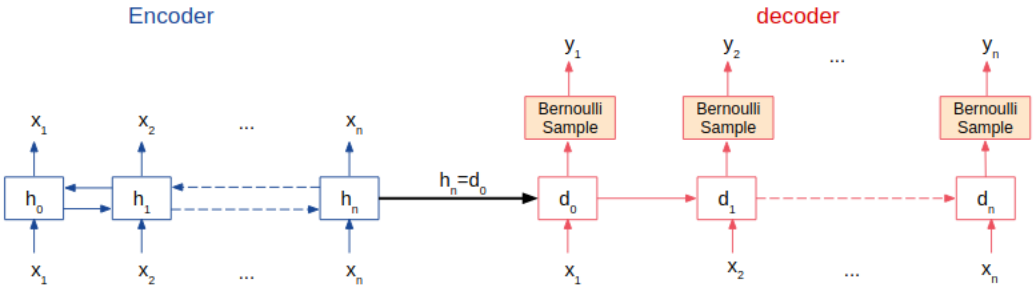

Figure 2: Network architecture of our translation model SMT+RL 1

As shown in Figure 2, the encoder is a bi-directional LSTM aiming to transform the input sequence w_x to its continuous representation h_n. Each hidden state h_t is represented by the forward and the backward hidden states $h_t = [\overrightarrow{h_t}; \overleftarrow{h_t}]$. Where $h_t \in R^{d_h}$ with d_h the number

of hidden dimensions. The hidden representations are defined as $\overleftarrow{h_t} = LSTM(\overleftarrow{h_{t-1}}, x_t)$ and $\overrightarrow{h_t} = LSTM(\overrightarrow{h_{t-1}}, x_t)$.

The decoder is composed of a LSTM layer in which each word x_i is injected to estimate the word selection probability $p(y_i|x_i, d_{i-1})$ using the hidden vector d_{i-1} and the current word x_i. The initial hidden state d_0 is initialized with the element-wise sum over the directions of the last encoder hidden state h_n.

- **SMT+RL 2:** In this variant, we only use a decoder with a binary memory.

$$d_t = \begin{cases} f_0(d_{t-1}, x_t, y_{t-1}), & \text{if } y_{-1} = 0 \\ f_1(d_{t-1}, x_t, y_{t-1}), & \text{if } y_{-1} = 1 \end{cases}$$

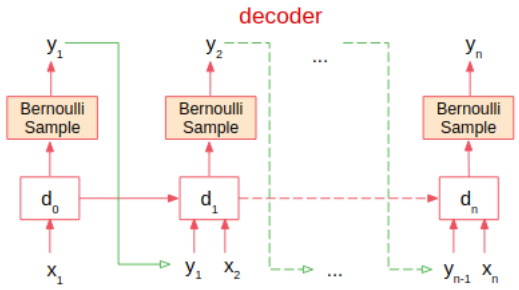

Figure 3: Network architecture of our translation model SMT+RL 2

The decoder is a LSTM layer in which each word x_i is injected to estimate the word selection probability conditioned by the previous word $p(y_i|x_i, d_{i-1}, y_{-1})$. We apply the function f_0 if $y_{-1} = 0$, and we use f_1 if $y_{-1} = 1$.

- **SMT+RL 3:** This model is a combination of the 2 previous ones, where we use an encoder-decoder with a binary memory.

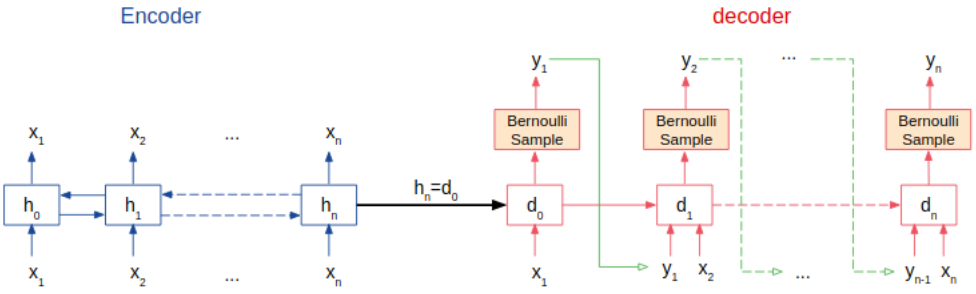

Figure 4: Network architecture of our translation model SMT+RL 3

- **SMT+RL 4:** In this case, we use the other type of memory, where we use two hierarchical LSTMs as explained above. In practice, the hidden states d_0 and s_0 of both LSTMs are initialized with the element-wise sum over the directions of the last encoder hidden state h_n. If a word is selected by the model, it is passed through the second level LSTM and its hidden state is added to the last hidden state of the first level LSTM.

$$d_0 = h_n$$
$$s_0 = h_n$$
$$d_t = \begin{cases} f_d(d_{t-1}, x_t, y_{t-1}), & \text{if } y_{t-1} = 0 \\ f_d(d_{t-1} + s_{t-1}, x_t, y_{t-1}), & \text{if } y_{t-1} = 1 \end{cases}$$
$$s_t = f_1(s_{t-1}, x_t, y_{t-1}), \text{if } y_t = 1$$

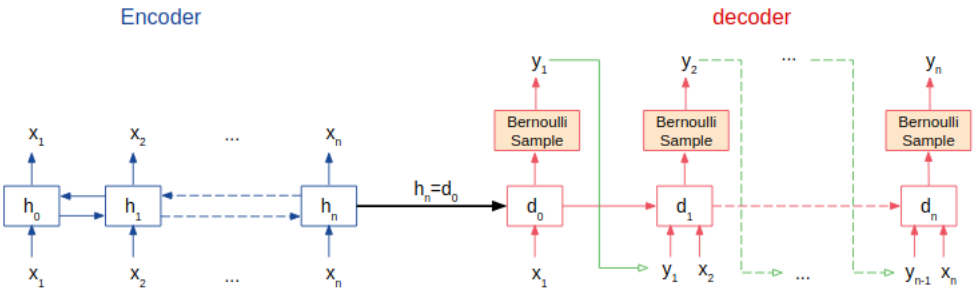

Figure 5: Network architecture of our translation model SMT+RL 4

- **model 5:** This model is similar to model 3 but at each decoding time step we also take as input the last hidden state of the encoder. As shown in 6, we feed a concatenation of x_i and h_n to a linear layer to get a shared representation x'_i and then use it as input for the decoder.

$$d_0 = h_n$$

$$x'_t = tanh(g([h_n; x_t]))$$

With g a linear function that takes the concatenation of h_n and x_t as input.

$$d_t = \begin{cases} f_0(d_{t-1}, x'_t, y_{t-1}, h_n), & \text{if } y_{-1} = 0 \\ f_1(d_{t-1}, x'_t, y_{t-1}, h_n), & \text{if } y_{-1} = 1 \end{cases}$$

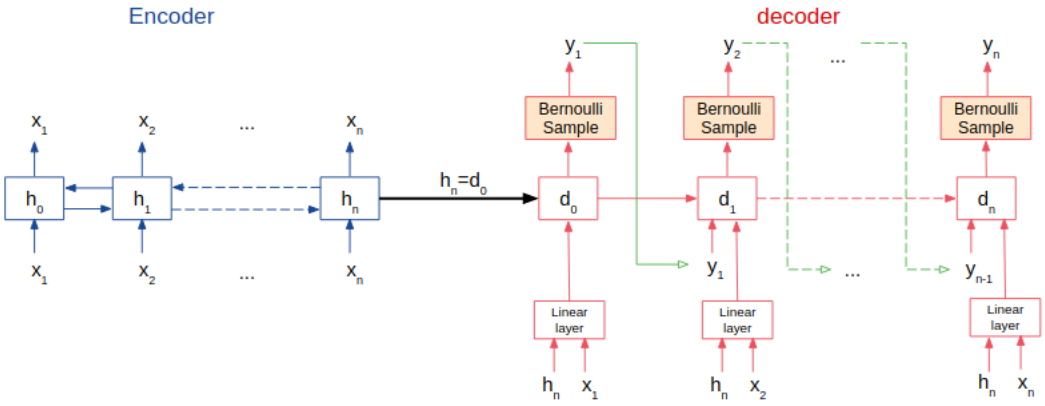

Figure 6: Network architecture of our translation model SMT+RL 5

3 Protocol Design

3.1 Datasets

Since there does not exist yet SOCS-driven datasets including NL-query pairs, we use TREC tracks (namely, Robust 2004 and Web 2000-2001). In these tracks, query topics include a title, a topic description and a narrative text; the two latter being formulated in natural language. To build query-NL pairs, we use the title to form the set of keyword queries and the description for the set of information needs expressed in NL. An example of a query-NL pair is:

Title	Lewis and Clark expedition
Description	What are some useful sites containing information about the historic Lewis and Clark expedition?

TREC track	collection	pairs	NL length	avg of duplic. word in NL
TREC Robust (2004)	disk4-5	250	15.333	1.108
TREC Web (2000 2001)	WT10G	100	11.47	0.65

Table 2: Dataset statistics separated per document collections

This NL-query building process results in 350 pairs in total as presented in Table 2.

We are aware that the use of TREC datasets is biased in the sense that it does not exactly fit with the expression of NL information need in the context of conversational systems, but we believe that the description is enough verbose to evaluate the impact of our query building model in this exploratory work. Further experiments with generated datasets, as done in (Song et al., 2017), will be carried out in the future.

We also analyze the issue of duplicate words into TREC descriptions since it can directly impact the query formulation process based on word selection in the word sequence of TREC descriptions. In practice, this might lead to select several times the same word to build the query, and, therefore, directly impact the retrieval performance. As shown in Table 2, the ratio of duplicate words in TREC descriptions over the whole set of queries is very low (1.1 duplicate words in average in each query for TREC Robust and 0.65 for TREC Web). This suggests that this issue is minor in the used datasets. We, therefore, decided to skip this issue for the moment.

3.2 METRICS AND BASELINES

To evaluate our approach, we measure the retrieval effectiveness of the predicted queries. To do so, for each predicted query, we run the BM25 model through an IR system (namely, PyLucene[1]) to obtain a document ranking. The latter is evaluated through the MAP metric.

To show the soundness of our approach (namely, translating information needs expressed in NL into queries), we compare our generated queries to scenario **NL** feeding the natural language information needs (TREC descriptions in our protocol) to the IR retrieval system.

Since the objective of our model is to formulate queries, we also evaluate the effectiveness of original TREC titles (scenario **Q**). This setting rather refers to the oracle that our model must reach.

We mentioned that before training the selection model we transformed each x to its binary representation y based on the presence of the words in the ground truth query. The dataset being slightly biased by this binary modeling, we observed that not all the words existing in the query do exist in x. To analyze this bias, we also compare our approach with these binary queries (scenario **Q bin**) referring to the projection of queries **Q** on the vocabulary available in the **NL** description.

We also compare our model to a random approach which randomly selects 3 words from x to build queries (scenario **Random**).

Different variants of our model are also tested:

- **SMT** which only considers the first component of our model based on a supervised machine translation approach (Section 2.2). This variant could be assimilated to the approach proposed in (Song et al., 2017) in the sense that the machine translation is performed independently of the task objective.
- **RL** which only considers the reinforcement learning objective function (Section 2.3) without pre-training of the supervised translation model.
- **SMT+RL 1, SMT+RL 2, SMT+RL 3, SMT+RL 4, SMT+RL 5** are our full models **SMT+RL** in which we start by pre-training the model using the supervised translation model (Section 2.3), and, then, we inject the reward signal in the translation probabilities (Section 2.4).

[1] `http://lucene.apache.org/pylucene/`

3.3 IMPLEMENTATION DETAILS

We implemented our framework in Python using Pytorch version 0.4. To transform each word x_i to its vector representation w_{x_i}, we use Fasttext [2] (Bojanowski et al., 2016) 300-dimensional pre-trained word embeddings. The encoder and decoders have one hidden layer with 100 hidden units each.

To train our model, we perform 10-fold cross-validation. At each round, we split the data into train and test sets such that each pair can be tested. For the 5 models, we start by a pre-training using the supervised translation until the model perfectly learns on trainset, we use the ground truth queries as targets for the training phase. The training is then pursued by 1000 iterations while including the reinforcement learning approach. In the latter, the reward, namely the MAP metric, is estimated over document rankings obtained by the BM25 model in PyLucene. We use a minibatch Adam (Kingma & Ba, 2014) algorithm to pre-train the model and SGD for the reinforcement learning part. Each update is computed after a minibatch of 12 sentences. During the decoding, we perform a beam search with beam size of 5. For the **SMT** and **RL** models we use the implementation of **SMT+RL 4**.

4 RESULTS

4.1 IMPACT OF THE NEURAL ARCHITECTURE

Baseline	TREC Robust(2004)	TREC Web (2000-2001)
SMT+RL 1	0.08020	0.15736
SMT+RL 2	0.07697	0.15882
SMT+RL 3	0.07219	0.15178
SMT+RL 4	**0.10286**	**0.17963**
SMT+RL 5	0.09951	0.16915

Table 3: The retrieval effectiveness (MAP) results of our five models on the datasets

In 3, we report the retrieval effectiveness (MAP) of our five proposed models (**SMT+RL 1, SMT+RL 2, SMT+RL 3, SMT+RL 4** and **SMT+RL 5**) detailed in section 2.4. We observe the following statements:

• Over the two datasets, **SMT+RL 4** achieves better results than the rest of the proposed models, we report a MAP of 0.09951 on TREC Robust(2004) and 0.16915 on TREC Web (2000-2001), we recall that in this model we encoded the NL information need with a Bi-LSTM encoder and then decoded it with a hierarchical decoder, this motivates our idea on the usefulness of initializing the decoder with a prior over the NL information need before selecting the words and supports the assumption that the decoder should have an extra memory of the words selected during the prediction.

• Also, **SMT+RL 5** yields better results than the rest of the proposed models (**SMT+RL 1, SMT+RL 2** and **SMT+RL 3**). This upholds the idea that the overall information need is more useful when injected to the model with each input when we use a binary memory rather than just use it to initialize the decoder in the case of a binary memory decoding. In fact, our **SMT+RL 3** (composed of an encoder and a decoder with a binary memory with $h_n = d_0$) is the least performing, we believe that this is due to the fact that the NL representation h_n may be annihilated by the sequential decoding and the use of the binary memory. This is mended by injecting the h_n at each time step with inputs x_i as done in **SMT+RL 5**, for instance, we can report an improvement of the MAP 0.09951 for **SMT+RL 5** against 0.07219 for **SMT+RL 3**.

• We can note that a hierarchical decoder (**SMT+RL 4**) for the already selected words is advantageous comparing to the binary memory (**SMT+RL 5**) or just using a simple decoder without keeping track of the previously select words (**SMT+RL 1**)

[2] https://github.com/facebookresearch/fastText/

4.2 RETRIEVAL EFFECTIVENESS OF OUR APPROACH

In the following we only consider our best proposed model **SMT+RL 4**.

Baseline	TREC Robust(2004)		TREC Web (2000-2001)	
	MAP	$\%Chg$	MAP	$\%Chg$
NL	0.08925	+15.25% ***	0.15913	+12.88% *
Q	0.09804	+4.92%	0.16543	+8.58%
Q bin	0.08847	+16.26% *	0.17402	+3.22%
Random	0.01808	+468.91% ***	0.04060	+342.44% ***
SMT	0.06845	+50.27% ***	0.08891	+102.04% ***
RL	0.08983	+14.51% ***	0.16474	+9.04%
SMT+RL 4	**0.10286**		**0.17963**	

Table 4: Comparative effectiveness analysis of our approach. $\%Chg$: improvement of **SMT+RL 4** (our best model) over corresponding baselines. Paired t-test significance *: $0.01 < t \leq 0.05$; **: $0.001 < t \leq 0.01$; ***: $t \leq 0.001$.

We present here the effectiveness of our approach aiming at generating queries from users' information needs expressed in NL. In Table 4, we present the retrieval effectiveness (regarding the MAP) of our model and the different baselines (**NL**, **Q**, **Q bin**, **Random**, **SMT**, and **RL**) described in section 3.2. From a general point of view, results highlight that in both datasets, our proposed **SMT+RL 4** outperforms the different baselines with improvements that are generally significant, ranging from $+3.22\%$ to $+468.91\%$.

NL	**Q**	**Q bin**	**SMT+RL**
what are new methods of producing steel	steel producing	producing steel	new methods of producing steel
what are the advantages and or disadvantages of tooth implant	implant dentistry	implant	advantages disadvantages tooth implant
find documents that discuss the toronto film festival awards	toronto film awards	toronto film awards	the toronto film festival awards
find documents that give growth rates of pine trees	where can i find growth rates for the pine trees	growth rates pine trees	growth rates of pine trees

Table 5: Examples of query formulation for **NL** queries, the original query **Q**, the binary version **Q bin** of the original query, and our model **SMT+RL 4**.

More particularly, the effectiveness analysis allows to draw the following statements:

• The overall performance of the compared approaches generally outperforms the retrieval effectiveness of the **NL** baseline. For instance, on TREC Robust, queries generated by our model allows to significantly improve the retrieval performance of $+15.25\%$ regarding information needs expressed in NL (MAP: 0.10286 vs. 0.08925). This result validates the motivation of this work to formulate queries from NL expressions. This is relatively intuitive since NL expressions are verbose by nature and might include non-specific words willing to inject noise in the retrieval process.

• Our approach **SMT+RL** provides similar results as the **Q** and **Q bin**. Since the objective function of our model is guided by the initial query **Q** transformed in a binary vector (**Q bin**), these baselines could be considered as oracles. We note however that our model obtains higher results (improvements from $+3.22\%$ to $+16.26\%$) with a significant difference for the **Q bin** baseline for TREC Robust. To get a better understanding to what extent our generated queries are different from those used in baselines **Q** and **Q bin**, we illustrate in Table 5 some examples. While queries in **Q** identify the most important words leading to an exploratory query (e.g. "steel productions"), our model **SMT+RL 4** provides additional words that precise which facet of the query is concerned (e.g., "new methods of..."), and accordingly improves the ranking of documents.

• Our model **SMT+RL 4** is significantly higher than the **SMT** baseline which converges to a relatively low MAP value (0.06845 and 0.08891 for TREC Robust and TREC Web, respectively). This could be explained by the fact that our datasets are very small (250 and 100 NL-query pairs respectively for TREC Robust and TREC Web) and that such machine translation approaches are

well-known to be data hungry. Reinforcement learning techniques could be a solution to overpass this problem since they inject additional information (namely, the reward) in the network learning.

• The **RL** baseline achieves relatively good retrieval performances. As we can see from TREC Web, the **RL** model obtains a MAP of 0.16474 against 0.15913 for the **NL** baseline. The **RL** baseline allows approaching the retrieval performances of baselines **Q** and **Q bin**, although it obtains lower results. This reinforces our intuition that 1) applying machine translation approaches should be driven by the task (retrieval task in our context) and 2) reinforcement learning techniques provide good strategies to build effective queries. The latter statement has also been outlined in previous work (Nogueira & Cho, 2017).

• The comparison of our models **SMT+RL** regarding **SMT** and **RL** baselines outlines that reinforcement learning techniques might be more beneficial when a pre-training is performed. In our context, the pre-training is performed using the **SMT** model (Section 2.3) which helps the model to be more general and effective before using the reward signal to guide the selection process.

It is worth mentioning that we also trained in preliminary experiments a state of the art translation models such as a generative encoder-decoder RNN with attention mechanism, as done in Yin et al. (2017); Song et al. (2017). We did not report the results since the model was not able to generalize in the testing phase over new samples from the NL-query dataset used in the training phase. This is probably due to the tradeoff between the number of training pairs and the large size of the vocabulary which is not enough represented in different contexts. However, we believe that combining reinforcement learning with attention-mechanism for query-generation is promising. We let this perspective for future work.

5 STATE OF THE ART

Chatbots are attracting increasing attention and recent contributions are based on deep learning techniques, neural networks are applied to solve the different components of dialogue systems including natural language understanding, dialogue management and response generation (Vinyals & Le, 2015). Among dialogue systems, two categories are distinguished: 1) Non task oriented systems also known as chit-chat bots (Yan et al., 2016), where the system is not meant to complete a specific task but exchange on open discussions. They generate responds based on the user's questions using generative models like sequence to sequence models. Other recent research in this vein has explored deep reinforcement learning techniques (Li et al., 2016b) and adversarial learning (Li et al., 2017) for neural dialogue generation. 2) task oriented systems (Bordes & Weston, 2016), which usually need to query knowledge bases to retrieve the needed informations where important facts are stored, for example, restaurant names that can be booked and their properties in case of a restaurant booking dialogue system. Search oriented systems (SOCS) belong to this second category.

Since users can express themselves freely using natural language expressions, natural language understanding (NLU) becomes extremely challenging for a chat-bot in order to capture the user's information need. In fact, NLU transforms a natural language utterance to its conceptual representation. Usually, the NLU task is divided into three sub-tasks: two classification tasks (namely the domain detection and the intention detection) and the slot filling sub-task (object of interest detection) (Tur, 2011). The applied approaches to tackle these tasks can be divided into three categories: i) rules based approaches (Campillos Llanos et al., 2015), ii) approaches based on data like statistical or neuronal approaches (Dinarelli et al., 2012), (Vukotic et al., 2015). iii) a combination of both (Bordes & Weston, 2016). These approaches provide good results but have different pros and cons. Data driven approaches require annotated data which are not always available while knowledge based approaches need an expert to develop the system. Recent work proved the importance of automatically generating annotated data from small amount of data or domain models (Bordes & Weston, 2016).

From an information retrieval point of view, search oriented conversational systems (SOCS) rely on the translation of the natural language information need to its key-words query form, this makes the retrieval task easier and more effective by reformulating the information need to a simpler and more precise sentence. Other than SOCS, information retrieval systems addressed the complex and long queries problem by relying on query suggestions to provide meaningful results and engaging experiences, this can be in the form of auto-completions or query reformulations. This makes the task of

language understanding easier by reformulating the information need to a simpler and more precise sentence. The work on query reformulation for web search depend on computer-generated suggestions using query expansion (Mitra et al., 1998), query substitution (Jones et al., 2006), and other refinement techniques (Kraft & Zien, 2004) (Baeza-Yates et al., 2005). Implicit relevance feedback from users is a common data source for computer-generated reformulations. For example, work by Baeza-Yates et al. (2005) uses query logs to discover new query reformulations, finding similar queries using a cosine function over a term-weighted vector built from the clicked documents. A study by Anick (2003) showed that these automatically generated reformulations were as effective as human constructed reformulations, using metrics such as uptake and click behavior.

Perhaps the closest work to ours is by Song et al. (2017), the authors raise the problem of lexical chasm between keyword queries and natural language queries: the retrieval performance of web search degrades when using a natural language query. The paper introduces a model to translate a natural language query into a keyword query. This model is an encoder-decoder RNN with attention mechanism in order to focus on the relevant words of the query and not only the whole semantics. The dataset contains 828,826 keyword query and natural language query pairs where the NL query is the the title of the top-1 retrived document using their search engin. The evaluation metrics used are NDGC@5 and 7 point-Likert. This approach achieved good results since the generated queries are three times shorter than the original ones and the useless terms are omitted by the translation process, thus, it learns only a mapping between the original query and the generated ones without taking into consideration the information retrieval task as we did in our work by optimizing the model to maximize the search effectiveness.

Another paper worth mentioning is Yin et al. (2017), in which the authors pointed the fact that question form queries are ambiguous and return less relevant results than key-word ones, the authors proposed a versatile framework for query understanding, ad recommendation and user interaction, it works in top of an existing information retrieval system. Given an input query in the form of a question, DeepProbe starts by rewriting it to keyword query (How to connect my tablet to tv $\rightarrow$ tablet tv connector) using an encoder-decoder with attention mechanism, then, relevant products are returned form the information retrieval system, if the confidence level is less than the threshold the framework asks questions to better understand the user need, these questions are based on the product attributes(size, color...). Finally the results are reordered by the relevance scorer and the system returns the most relevant results to the user. Our method is more general because we don't only focus on question form queries.

In this context, reinforcement learning (Sutton et al., 1999) is gaining traction in natural language understanding across many problems, recent works started to investigate on reinforcement learning techniques in sequence prediction other than just using cross-entropy loss, this was motivated by the fact that they alleviate limitations inherent to the word-level optimization of the cross-entropy loss, allowing the use of sequence-level reward functions. This was applyed in dialogue generation (Li et al., 2016b), question generation from text (Kumar et al., 2018), query reformulation (Nogueira & Cho, 2017), active question reformulation (Buck et al., 2017).

Recent work addressed the query reformulation task by applying reinforcement learning to understand the actual user intent as Nogueira & Cho (2017), Nogueira et al. proposed a reinforcement learning framework to reformulate queries in order to retrieve more relevant documents. The initial user's query q_0 is used to retrieve the set of documents D_0 from a search engine. The formulator generates a new query q' by selecting terms from the union of q_0 and D_0, then, the new set of documents D' is retrieved using q'. The document recall is computed over the D' and is used as the reward signal to train the model. The authors test both a CNN and a RNN to convert the query sentences into vector representations and for each candidate term they transform it into its vector representation along with its context words. The resulting two vector representations are concatenated and the probability of selecting a candidate term is computed by two feed forward layers. The proposed approach outperforms strong baselines on three different datasets and the evaluation metrics are R@40, P@10 and MAP@40. However, the number of candidate terms is extremely large, the authors test the model on the maximum number of terms they could fit in a GPU, hence, they take the top 7 retrieved documents based on q_0 and run the model over the 300 first words of each document. At training time of the RL model, they only sample one document from the top 7 retrieved documents as a source for candidate terms. This approach leads to a faster training but may prevent the model from discovering better formulation strategies. In contrast, our method

doesn't require new words in the selection process, we only select the most important words from the original query.

In this work, we combine machine translation models to translate a NL information need to its keyword query leaned on 2 different TREC datasets and then augment the model with a reinforcement learning reward signal to predict more effective key-words.

6 Conclusion and future work

We propose a selection model to transform user's need in NL into a keyword query to increase the retrieval effectiveness in a SOCS context. Our model bridges two lines of work dealing with supervised machine translation and reinforcement learning. We reframed a generation problem as a selection one and used a reward signal to leverage the small available dataset size problem. We implemented five variants of our models based on different assumptions and concluded that combining an encoder with an hierarchical decoder is the best combination. Our models has been evaluated using two different TREC datasets and outlines promising results in terms of effectiveness. Our approach has some limitations we plan to overcome in the future. First, our model is framed as a word selection process that could be turned into a generative model. Second, experiments are carried out on small datasets (250 and 100 NL-query pairs) that could be augmented using the evaluation protocol proposed in Song et al. (2017). In long term, we plan to adapt our model by totally skipping the query formulation step and designing retrieval models dealing with NL expressions.

References

Eugene Agichtein, Eric Brill, and Susan Dumais. Improving web search ranking by incorporating user behavior information. In *SIGIR '06*, pp. 19–26, 2006.

Wafa Aissa, Laure Soulier, and Ludovic Denoyer. A reinforcement learning-driven translation model for search-oriented conversational systems. In *SCAI Workshop - EMNLP*, 2018.

Peter Anick. Using terminological feedback for web search refinement: A log-based study. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, SIGIR '03, pp. 88–95, 2003. ISBN 1-58113-646-3.

Ricardo Baeza-Yates, Carlos Hurtado, and Marcelo Mendoza. Query recommendation using query logs in search engines. In Wolfgang Lindner, Marco Mesiti, Can Türker, Yannis Tzitzikas, and Athena I. Vakali (eds.), *Current Trends in Database Technology - EDBT 2004 Workshops*, 2005.

Jonathan Baxter, Lex Weaver, and Peter Bartlett. Direct gradient-based reinforcement learning: Ii. gradient ascent algorithms and experiments. Technical report, National University, 1999.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information, 2016.

Antoine Bordes and Jason Weston. Learning end-to-end goal-oriented dialog. *CoRR*, abs/1605.07683, 2016.

Christian Buck, Jannis Bulian, Massimiliano Ciaramita, Wojciech Gajewski, Andrea Gesmundo, Neil Houlsby, and Wei Wang. Ask the right questions: Active question reformulation with reinforcement learning. 2017.

Mikhail Burtsev, Aleksandr Chuklin, Julia Kiseleva, and Alexey Borisov. Search-oriented conversational ai (scai). In *ICTIR '17*, pp. 333–334. ACM, 2017.

Leonardo Campillos Llanos, Dhouha Bouamor, Éric Bilinski, Anne-Laure Ligozat, Pierre Zweigenbaum, and Sophie Rosset. Description of the patientgenesys dialogue system. In *Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL 2015)*, 2015.

Bhuwan Dhingra, Lihong Li, Xiujun Li, Jianfeng Gao, Yun-Nung Chen, Faisal Ahmed, and Li Deng. Towards end-to-end reinforcement learning of dialogue agents for information access. In *ACL' 17*, pp. 484–495, 2017.

M. Dinarelli, A. Moschitti, and G. Riccardi. Discriminative reranking for spoken language understanding. *IEEE Transactions on Audio, Speech, and Language Processing*, 2012.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S. Weld. Knowledge-based weak supervision for information extraction of overlapping relations. In *HLT '11*, pp. 541–550, 2011.

Thorsten Joachims. Optimizing Search Engines Using Clickthrough Data. In *SIGKDD '02*, pp. 133–142. ACM, 2002.

Hideo Joho, Lawrence Cavedon, Jaime Arguello, Milad Shokouhi, and Filip Radlinski. Cair'17: First international workshop on conversational approaches to information retrieval at sigir 2017. *SIGIR Forum*, 51(3):114–121, 2018.

Rosie Jones, Benjamin Rey, Omid Madani, and Wiley Greiner. Generating query substitutions. In *Proceedings of the 15th International Conference on World Wide Web*, WWW '06, 2006.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

Reiner Kraft and Jason Zien. Mining anchor text for query refinement. In *Proceedings of the 13th International Conference on World Wide Web*, WWW '04, pp. 666–674, 2004. ISBN 1-58113-844-X.

Vishwajeet Kumar, Ganesh Ramakrishnan, and Yuan-Fang Li. A framework for automatic question generation from text using deep reinforcement learning, 2018.

Guillaume Lample, Ludovic Denoyer, and Marc'Aurelio Ranzato. Unsupervised machine translation using monolingual corpora only. *CoRR*, abs/1711.00043, 2017.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. A diversity-promoting objective function for neural conversation models. In *HLT '16*, pp. 110–119. ACL, 2016a.

Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. Deep reinforcement learning for dialogue generation. *CoRR*, abs/1606.01541, 2016b.

Jiwei Li, Will Monroe, Tianlin Shi, Sbastien Jean, Alan Ritter, and Dan Jurafsky. Adversarial learning for neural dialogue generation. In Martha Palmer, Rebecca Hwa, and Sebastian Riedel (eds.), *EMNLP*, pp. 2157–2169. Association for Computational Linguistics, 2017. ISBN 978-1-945626-83-8.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*, pp. 2181–2187. AAAI Press, 2015.

Mandar Mitra, Amit Singhal, and Chris Buckley. Improving automatic query expansion. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '98, 1998.

Rodrigo Nogueira and Kyunghyun Cho. Task-oriented query reformulation with reinforcement learning. In *SCAI Workshop - ICTIR*, 2017.

Alan Ritter, Colin Cherry, and William B. Dolan. Data-driven response generation in social media. In *EMNLP '11*, 2011.

Hyun-Je Song, A-Yeong Kim, and Seong-Bae Park. Translation of natural language query into keyword query using a rnn encoder-decoder. In *SIGIR '17*, pp. 965–968, 2017.

Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Proceedings of the 12th International Conference on Neural Information Processing Systems*, NIPS'99, pp. 1057–1063, Cambridge, MA, USA, 1999. MIT Press.

Gokhan Tur. *Spoken Language Understanding: Systems for Extracting Semantic Information from Speech*. John Wiley and Sons, January 2011.

Svitlana Vakulenko, Ilya Markov, and Maarten de Rijke. Conversational exploratory search via interactive storytelling. In *NEUIR SIGIR'17*, 2017.

Oriol Vinyals and Quoc Le. A neural conversational model, 2015.

Vedran Vukotic, Christian Raymond, and Guillaume Gravier. Is it time to switch to Word Embedding and Recurrent Neural Networks for Spoken Language Understanding? In *InterSpeech*, Dresde, Germany, 2015.

Zhuoran Wang and Oliver Lemon. A simple and generic belief tracking mechanism for the dialog state tracking challenge: On the believability of observed information. In *SIGDIAL' 13*, pp. 423–432, 2013.

Rui Yan, Yiping Song, and Hua Wu. Learning to respond with deep neural networks for retrieval-based human-computer conversation system. In *ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '16, pp. 55–64, 2016.

Zi Yin, Keng-hao Chang, and Ruofei Zhang. Deepprobe: Information directed sequence understanding and chatbot design via recurrent neural networks. In *SIGKDD' 17*, pp. 2131–2139, 2017.