Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020), pages 5474–5480
Marseille, 11–16 May 2020

Chat or Learn: a Data-Driven Robust Question-Answering System

Gabriel Luthier and Andrei Popescu-Belis
HEIG-VD / HES-SO
Route de Cheseaux 1, CP 521
1401 Yverdon-les-Bains, Switzerland
{gabriel.luthier, andrei.popescu-belis}@heig-vd.ch

Abstract

We present a voice-based conversational agent which combines the robustness of chatbots and the utility of question answering (QA) systems. Indeed, while data-driven chatbots are typically user-friendly but not goal-oriented, QA systems tend to perform poorly at chitchat. The proposed chatbot relies on a controller which performs dialogue act classification and feeds user input either to a sequence-to-sequence chatbot or to a QA system. The resulting chatbot is a spoken QA application for the Google Home smart speaker. The system is endowed with general-domain knowledge from Wikipedia articles and uses coreference resolution to detect relatedness between questions. We present our choices of data sets for training and testing the components, and present the experimental results that helped us optimize the parameters of the chatbot. In particular, we discuss the appropriateness of using the SQuAD dataset for evaluating end-to-end QA, in the light of our system's behavior.

Keywords: question answering, dialogue acts, chatbots, SQuAD, Wikipedia, movie subtitles.

1. Introduction

The multiplication of voice-based virtual assistants available on mobile phones or smart speakers has made conversational systems familiar to the general public. Commercial assistants from large technology companies focused initially on tasks related to the main offerings of these companies: controlling a smartphone (Apple's Siri), purchasing goods or services (Amazon's Alexa), or searching for information over the Web (Google Assistant). Beyond these functionalities, which can in principle be achieved with large finite-state dialogue models, commercial assistants were not intended to support extended conversations on topics in the general domain, although their capacities for chitchat have been gradually increasing (Fang et al., 2018). On the contrary, the systems known as chatbots were intended from the start to offer robust conversational capacities, although with a thin and often implicit knowledge base. Such chatbots are often trained on large dialogue corpora from movies or social media. However, giving them task-oriented capabilities is difficult, at least when using the sequence-to-sequence neural models on which most recent chatbots rely.

In this paper, we present a simple yet effective solution to design a conversational agent that combines the robustness of chatbots with the utility of task-oriented systems – here, question answering (QA). The system supports task-oriented dialogues, to answer factual questions over a document set such as Wikipedia, but has also the capacity to reply to non-task-oriented questions, for managing the social aspects of the conversation including chitchat (or small talk). Ahead of the two components supporting these functionalities, which are implemented using state-of-the-art technologies, the system relies on a controller, which performs dialogue act recognition and routes user utterances to one of the components. The system operates using the Google Home smart speaker for speech-based interaction, or using a text-based interface.

The paper is organized as follows. We first review the main existing technologies and the limitations we attempt to address with our architecture (Section 2). Then, we present the main design and implementation choices for our system (Section 3): question answering, including the management of follow-up questions thanks to pronoun resolution, chitchat with sequence-to-sequence models, and dialogue control using dialog act recognition. Evaluation results are given in Section 4 for the three components. We examine in particular the relevance of the SQuAD data set for testing document-based end-to-end QA, as opposed to answer extraction from paragraphs only, and discuss the data and parameters required to train the components for best performance.

2. Motivation

Task-oriented dialogue systems have been extensively studied in the past decades (Rieser and Lemon, 2011), but despite significant achievements in their training, the design or adaptation of such systems remains very costly. This is mostly due to their highly structured nature, with dedicated modules for language processing, dialogue modeling, and task management. This is the case, for instance, when the dialogue management component relies on explicit finite-state representations of dialogue states, with dialogue policies based on Partially Observable Markov Decision Processes (POMDP); such models must be trained with reinforcement learning, possibly using simulations of human users (Rieser and Lemon, 2011, Chapter 3).

The advent of deep learning has opened new possibilities for end-to-end training of conversational agents, typically with neural sequence-to-sequence models (Li et al., 2016). This is mainly successful for generating fluent and plausible responses when chatting, and has been essentially applied to chatbots for non-task-oriented dialogues, such as Cleverbot.com or TockTock (Zhou et al., 2016).

Dhingra et al. (2017) have proposed a solution for end-to-end training of a recurrent neural network intended for task-oriented dialogue, by converting probabilities in the final

layer of the network into discrete entities, and thus allowing a movie database to be queried. However, the system did not generate utterances and its task was limited to attribute-based search in a data table.

Several large technology companies have designed robust smart assistants and put them in production. Such assistants rely on cloud-based solutions for speech recognition and synthesis, and on undisclosed dialogue management architectures, likely inspired from finite-state models. These assistants were initially tailored to offer services related to the main business of their provider (e-commerce, control of smartphone and apps, or Web search), although recent extensions have enabled some of them to support several turns of chitchat. At least two providers enable the connection of third-party systems to their smart speakers (Google Home and Amazon Echo) while another one supports written interactive applications (Facebook). To this end, specific frameworks offer basic NLP functions and models for defining dialogue policies: the Alexa Skills Kit (for Amazon Echo), DialogFlow (connected to Google Home via Actions), or WIT.AI (related to Facebook Messenger).[1]

Traditional QA systems operate in two phases (Harabagiu et al., 2003). First, they identify the document or paragraph that potentially contains the answer to the given question, typically using methods pertaining to information retrieval. Second, they extract the specific answer using NLP techniques, relying in particular on the estimation of the type of the expected answer. Interactive QA systems are more challenging to design than non-interactive (single turn) ones, as dialogue states and policies are difficult to define in the case of open questions (Rieser and Lemon, 2009).

Neural systems have recently demonstrated impressive QA capabilities (Chen et al., 2017), more specifically on the second part of the QA task, i.e. *extracting answers given a question and a paragraph*. Recent progress has been made using BERT, i.e. Bidirectional Encoder Representations from Transformers (Devlin et al., 2018). For QA, or more exactly for answer extraction, the BERT model is post-trained with a final layer dedicated to this task, and various systems using it with various strategies, including ensemble models, have recently reached nearly human-like levels on the SQuAD benchmark (Rajpurkar et al., 2016; Rajpurkar et al., 2018).[2]

SQuAD 1.1 contains 100,000 questions derived by human workers from about 500 paragraphs from Wikipedia. Each question is accompanied by the correct answer, and by the paragraph on which it is based. It is thus possible to test answer extraction, but also end-to-end QA by hiding the paragraphs, though this method has several limitations (discussed in Section 4.1 below). SQuAD 2.0 adds 50,000 "unanswerable" questions, which systems should tag as such. Their role is to penalize systems that would aggressively search for answers in paragraphs. Our system augments with paragraph retrieval a system based on BERT, and uses SQuAD scores as a criterion to tune the system's parameters.

[1] `developer.amazon.com/alexa-skills-kit`, `wit.ai`, `dialogflow.com`, `developers.google.com/actions`.

[2] `rajpurkar.github.io/SQuAD-explorer/`

Finally, in our system, the controller that conveys utterances either to the chatbot or to the QA component uses dialogue act recognition, which we envisage here as a simple classification task. More complex sequence models (Stolcke et al., 2000) as well as neural models (Zhao and Kawahara, 2017) have been proposed and could be used in the future to improve performance.

3. Design and Implementation

The main components of our conversational agent are shown in Figure 1. The input utterance obtained from a smart speaker with speech recognition or from a text-based interface is passed to the dialogue act recognizer. This component routes task-related inputs (i.e. genuine information-seeking questions) to the QA component, and other inputs (including greetings, politeness or non-goal-directed chat) to the chatbot. The answers from the respective component are then sent back to the smart speaker using speech synthesis or to the text-based interface.

This structure offers a simple solution for designing a robust agent for accessing information that can be found in documents, and reduces the risk of failure when presented with non-information-seeking questions. Although these components are reminiscent of the traditional ones – respectively dialogue management, task management, and user interaction – they are organized in a quite different manner.

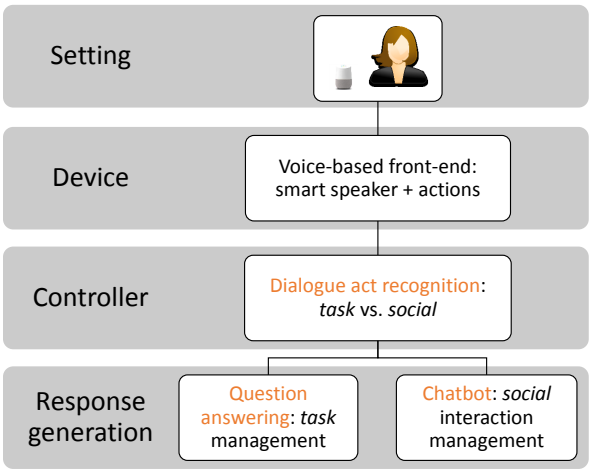

Figure 1: Architecture of our conversational agent.

3.1. Question Answering Component

While our agent is intended as a general solution for mixing task-oriented dialogue with chatting, in its current version the task under consideration is specifically a QA one, assuming answers to a user's questions can be found in the English Wikipedia pages. Given the outstanding recent results of the BERT model applied to QA on the SQuAD dataset, we selected this model for our QA component. However, the model is designed for answer extraction from a specific paragraph – with provision for the cases when the paragraph does not contain the answer – while we need to develop an end-to-end solution, starting directly from the

document repository, without prior knowledge of the paragraphs relevant to each question.

To retrieve from the document set the paragraphs that are likely to contain the answer to a given question, we use the Elasticsearch information retrieval system,[3] itself based on Apache Lucene. The document set providing knowledge to our system is made of Wikipedia pages, which we retrieve from a Wikipedia dump in an appropriate format for Elasticsearch, named Cirrussearch.[4] Given a question, the system retrieves from Wikipedia one or more relevant paragraphs, transforming the question into a query, with the parameters presented below. The paragraphs are then processed by the answer extraction component using BERT.

3.1.1. Design Options

For integrating paragraph search (ElasticSearch) with answer extraction (BERT), we experimented with various combinations of the following options.

Question preprocessing: filter out stop words or not.

Elasticsearch: perform the search either over the full text of each Wikipedia page, or over the opening paragraph(s) only (i.e. the initial text before the first section, which is stored in a specific field in the Cirrussearch Wikipedia dump), or over the title only.

Search results: keep 1-best or up to 5-best results; in the latter case, the answer extraction method examines the results by decreasing Elasticsearch scores and stops as soon as it finds an answer.

Answer extraction with BERT: applied to the opening text of the page, or to the entire page.

We selected the optimal options based on the SQuAD scores shown in Table 1 of Section 4– we also discuss the impact of unanswerable questions in Section 4. The best options we found are: (a) to consider the *unfiltered questions*; (b) to search with Elasticsearch only over the *titles* of Wikipedia pages; and (c) to give BERT only the *opening paragraphs* of the *1-best* result.

We do not perform any other form of fine-tuning of BERT-base to our task, as we do not have a more specific question set than SQuAD at this stage of the project.

3.1.2. Pronoun Resolution

Series of questions targeting the same entity are fairly common, and in this case pronouns are typically used to refer to the entity on subsequent mentions, rather than repeating a full description of it. We handle this phenomenon within the QA component only, rather than across the two components (QA and chatbot). We assume that if an utterance was classified as chitchat, co-reference is no longer possible across it, as topics may change quite unpredictably.

We incorporated the NeuralCoref pronoun resolution module[5] to the QA component, making it capable of handling follow-up questions. When a third person pronoun (personal or possessive) is detected in a question from the user, NeuralCoref is called on the utterance, with the latest two question-answer pairs as context. If NeuralCoref finds an antecedent, then our system replaces the pronoun in the user's question with this antecedent, and passes the resulting question to Elasticsearch and then BERT. The substitution of a pronoun by its antecedent may degrade the fluency of the question, but the substituted result is only used by the search components, and is never seen by the user. This use of pronoun resolution keeps it separate from the search component, enabling independent experiments with each component. Such a simple logic can accommodate other knowledge sources without changing the search algorithm.

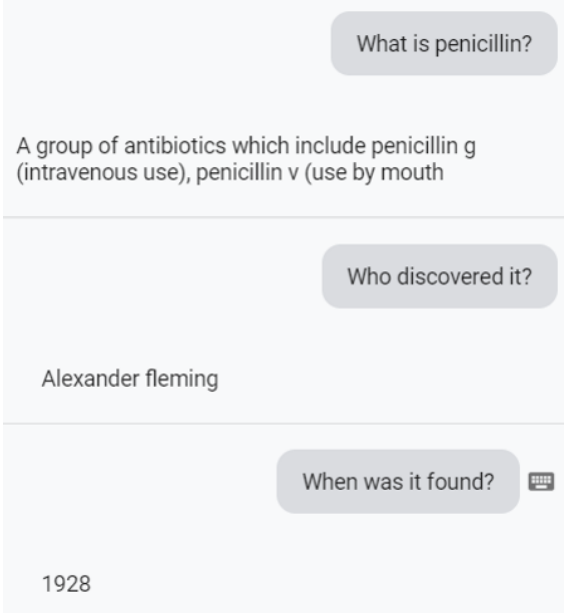

Figure 2: Sample conversation with the agent.

A sample text-based conversation with three information-seeking questions is shown in Figure 2. In conditions with moderate audio noise, the Google Home device is able to recognize this input without errors. The pronouns appearing in the second and third questions are correctly solved, and the answers are correctly found, though the first one appears to be imperfectly truncated by BERT from a longer Wikipedia sentence.

3.2. Chatbot Component

The chatbot component is designed to answer the user's questions that are not requests for information, without attempting to perform a search, but providing a fluent and socially acceptable reply. The utterances routed to this component include dialogue openings and closings, politeness-related utterances, and more generally any utterances which may come from users who want to play with the system, possibly for entertainment, by pushing it off-task. Several types of non-task utterances can be recognized by the DialogFlow toolkit to which the Google Assistant is connected, but we do not rely on this functionality, as we do not want to tie our system strongly tied to this framework.

To handle non-task-oriented utterances with a robust and unified mechanism, without investing a large effort in processing each sub-type of off-task utterance, we decided to

[3] www.elastic.co

[4] dumps.wikimedia.org/other

[5] github.com/huggingface/neuralcoref

train a neural sequence-to-sequence model based on recurrent neural networks with LSTMs, proposed by Vinyals and Le (2015) and for which a PyTorch implementation is available.[6] A second implementation of the same model, provided by Yuan-Kuei Wu,[7] has also been tested for comparison purposes. As already observed by Vinyals and Le (2015), this seq2seq model cannot ensure consistency across several turns, but here it allows rapid development of a system for handling non-task-related utterances.
The chatbot component was trained over a small subset (the first 25 MB of a total of 25 GB) of the English part of the OpenSubtitles corpus,[8] with the set of subtitles for each movie counting as one dialogue.

3.3. Dialogue Controller

Our proposal for switching between the chatbot and the QA components relies on a dialogue act recognizer implemented as an utterance classifier. Its goal is to distinguish the requests for information, which are sent to the QA system, from all other utterances including greetings and chitchat, which are sent to the chatbot. Therefore, instead of a using a complex dialogue act hierarchy (Bunt et al., 2012), our controller only needs to make a binary decision, which can be carried out using context-free classifiers (Clark and Popescu-Belis, 2004). For this, it uses an utterance classifier that is trained on a labeled data set constructed by joining SQuAD questions and movie dialogues (about 10,000 lines of each type), which all come directly from the original training data of each component.

3.4. Software and Hardware Platforms

We use the Google Home smart speaker as a speech-based front-end to our conversational agent. Agents called 'Actions' can be created on a platform dedicated to developers,[9] which is wired to the DialogFlow platform offering several out-of-the-box NLP tools. It is also possible to connect an Action to a remote server which can handle the dialogue, as our system does to a server located in our institute. Our server is an endpoint that reads the payload containing the user's input words, triggers the conversational agent, and sends back the output words, which are then uttered by the smart speaker. The entire computation of the answer given an utterance is performed on our local server with a high-end single GPU board. In practice, the observed time interval between a user's utterance and the reply is acceptable for natural spoken interaction (0.5–2 seconds).

4. Evaluation

The end-to-end evaluation of our system is nontrivial. A theoretical option, which comes at a high cost, is to define a context of use, select a specific document repository, and ask users to test the system. Another option is to use a large set of questions that jointly test the performance of the controller, the QA component and the chatbot – however, to the best of our knowledge, such a data set does not exist. Therefore, we use several component-level evaluation methods along with our own subjective testing, to assess the overall system's performance.

4.1. Evaluating QA with SQuAD: Results and Limitations

We optimized several parameters of the QA component by attempting to increase its scores on the SQuAD devset. The options described in Section 3.1 above have been evaluated, and the obtained scores are presented in Table 1. The analysis of the scores enabled us to make several design choices for our component.

4.1.1. Global Scores

We use the same metric as the one from the official SQuAD website (see footnote 2), namely Exact Match, i.e. "the percentage of predictions that match any one of the ground truth answers exactly" (Rajpurkar et al., 2016). In our case, the paragraph from which BERT extracts the answer is not provided with the question, but must be found by searching Wikipedia. This explains the lower scores of our system, which answers correctly only about half of the SQuAD 2.0 questions.
To quantify the role of Elasticsearch vs. BERT-base, we also evaluated locally the BERT-base stage using the paragraphs provided with SQuAD, and obtained 72.5% accuracy. This is similar to published BERT-base scores, though it is considerably lower than systems using the full BERT model or other language representation models, as well as ensembles of models, which have very recently reached the 90% mark.
We infer that the main weakness of our method is the paragraph retrieval component, and confirm this below by examining separately the performance on SQuAD questions that have an answer vs. the performance on the "unanswerable" questions (5,928 versus 5,945, for a total of 11,873), as shown in Table 2. We relaxed the Exact Match metric to allow for case-independent matching and a Levenshtein distance of at most 3. While our system identifies correctly a large proportion of unanswerable questions, it fails to find answers to many questions that actually have an answer in the SQuAD paragraph.

4.1.2. Scores for Answerable Questions

The detailed analysis of our system's performance on answerable questions, in the upper part of Table 2, shows that for our correct as well as incorrect answers, only a small proportion of the actual SQuAD paragraphs were retrieved: 19 out of 300 for questions answered correctly (6.3%), and 26 out of 1,614 (1.6%) for those answered incorrectly. Moreover, for a very large proportion of these questions (3,924 out of 5,928 i.e. 66%), no page was retrieved at all by Elasticsearch.
One reason explaining this finding is that many SQuAD questions target minor facts in the source paragraph (e.g. "Who was Robert's son?"or "Who kicked Ethelred out?") or are highly non-specific (e.g. " What are two examples of different types of reduction?" or "Political disadvantage is an attribute of which state policies?") Hence, when we

[6] pytorch.org/tutorials/beginner/chatbot_tutorial.html
[7] github.com/ywk991112/pytorch-chatbot
[8] opus.nlpl.eu/OpenSubtitles-v2018.php
[9] developers.google.com/actions

		Raw questions			Filtered questions		
		Search on titles	Search on openings	Search on full texts	Search on titles	Search on openings	Search on full texts
BERT on openings	1st answer	**47.8**	46.1	46.1	46.6	42.8	46.3
	5 answers	42.5	38.6	39.6	38.1	32.7	34.2
BERT on full texts	1st answer	45.1	41.4	40.4	39.2	38.2	37.6
	5 answers	39.1	30.2	27.9	28.0	24.7	23.5

Table 1: Accuracy of QA for SQuAD 2.0 exact matches, *without* using the paragraph provided for each question.

Reference	**Our system**		
	Correct answers	**Incorrect answers**	**No answer (no page)**
Answerable	**300**	1,614	3,924
· *same wiki page*	*19*	*26*	
· *different page*	*281*	*1,598*	
Unanswerable	N/A	1,115	**4,830**
· *same wiki page*		*31*	
· *different page*		*1,084*	

Table 2: Analysis of end-to-end question answering with SQuAD 2.0 questions, *without* using the paragraph provided for each question. Correct cases are those in bold numbers.

use such questions as queries, it is impossible for Elasticsearch to retrieve the paragraph from which they were actually constructed. This points to an intrinsic weakness of SQuAD for end-to-end QA evaluation.

In fact, most of the correct answers scored by our system were actually found in different Wikipedia paragraphs than the source ones in SQuAD. This shows that our method is able to extract knowledge if it is available somewhere in the document repository. Our method sometimes even provides better answers than those found in SQuAD. For instance, for the question "How many campuses does the University of California have?", our method finds the correct answer (10) from the Wikipedia page on the 'University of California', while SQuAD indicates a wrong answer (5) from a paragraph on 'Southern California'. Assessing the number of wrong answers in SQuAD should be a priority for the future.

A final issue with SQuAD was detected by examining the paragraphs that our system correctly found, but from which the answer was not extracted correctly by our version of BERT (26 cases). While some mistakes are due to BERT, most of the problems come from the fact that some articles are substantially different between SQuAD and our Wikipedia dump (footnote 4). This was the case for 63 SQuAD questions based on 'Warsaw' and 27 based on 'Rhine', most of which were absolutely not answerable and for this reasons are removed from the counts in Table 2. The other articles in this case were: 'Imperialism' (15 questions), 'Ctenophora' (11), 'Oxygen' (7), 'Normans' (3), 'Islamism' (2), and 'Construction' (1), which we all kept in our statistics because they still contained the correct answers despite substantial changes in the Wikipedia text. The wrong answers for these paragraphs are simply due to BERT mistakes.

4.1.3. Scores for Unanswerable Questions

Interpreting our scores on unanswerable questions is more difficult. For a very large proportion of these questions, Elasticsearch finds no relevant paragraph, and thus our system's empty answer is scored as correct. However, when actually finding a paragraph and an answer (1,115 questions out of the 5,945 unanswerable ones), our system sometimes finds a correct answer from a different paragraph, but this doesn't increase its SQuAD score as the question is considered unanswerable from its original paragraph.

For instance, the question "What is France a region of?" cannot be answered from the SQuAD paragraph on 'Normandy', but our system provides a correct answer from the opening text of the page on 'France', namely "Western Europe". Similarly, our system answers correctly the question "Who married Cnut the Great?", which is unanswerable from 'Normans' but is answered based on "Cultural depictions of Cnut the Great", with the correct answer 'Emma of Normandy'. Of course, none of these contribute positively to our SQuAD score, as they are marked as unanswerable in SQuAD.

Finally, some unanswerable questions are so poorly formulated that they rule out the possibility of finding an answer even if they had one in the data set. Some examples are: "when did Nors encampments ivolve into destructive incursions?" (where 'Nors' and 'ivolve' are misspelled) or "What is Las Vegas one of in the United States?" or "Whose value was know in Newton's life?"

4.2. Evaluating the Seq2seq Component

For the chatbot component, evaluation is harder since it cannot be easily automated, while user-oriented evaluation is beyond our resources and scope. Hence, in order to get a sense of the system's behavior, we tested the system with around twenty elementary utterances – such as "Hello", "Good bye", "Are you a robot?", "What's your favourite food?" – for several values of a variety of parameters. We tested the two implementations of the sequence-to-sequence model presented above, and the first one (from the PyTorch tutorial) appeared to outperform the second one. Although we noticed some fluctuation of the replies, the different parameters for training and testing both implementations did not lead to systematic improvements or degradations. The main problem was the generation of replies that seemed unrelated to the input utterance, as well as the rather low variations of the answers' quality with the parameters.

Other training sets were tested, such as the Cornell Movie-

Dialogs Corpus[10] and the dialogue section of the British National Corpus[11] which led to more perceptible changes than when modifying the chatbot's parameters, but not sufficient to select these as final training sets.

4.3. Performance of the Dialogue Controller

The evaluation of the dialogue controller component was made using held-out data. Our data set contained the same amount of SQuAD questions and of movie dialogues: 11,873 lines of each type. We randomly divided the corpus into training, validation and test subsets (80%, 10% and 10%) and used the test subset to assess the performance of the model. The classifier reached 90% accuracy, and observation of its predictions while using the system confirmed this high quality.

5. Conclusion and Future Work

We presented a conversational agent implemented as an Action on Google Home, which can answer questions in the general domain based on Wikipedia, but can also manage greetings and chitchat thanks to the integration of two dedicated components. Interaction with the system appears to be reasonably fluent and entertaining, as long as the user maintains her initiative and interest in the interaction.

We intend in fact to use this system as front-end to corporate information available in documents, which will require us to design a self-contained system with fully-controlled speech recognition and synthesis, as in our prototype of document recommender for conversations (Popescu-Belis et al., 2011). One project under way is to embody it into a humanoid robot such as Aldebaran's Nao or Pepper. Evaluation methods will also change with the domain, the documents, and the hardware.

A weakness of the dialogue controller implemented as a binary classifier is it difficulty to handle follow-up utterances such as requests for clarification. A way to address the issue is to train the classifier to recognize more types of dialogue acts, such as requests for answer refinement, indications of misunderstanding, or requests for additional answers (Bunt et al., 2012). Another strategy is to submit each utterance to both components, and learn to decide which answer to show based on the question and the two answers.

Expanding the system to other languages is also part of our future plans. The main difficulty resides in fine-tuning a language-specific BERT model with a SQuAD-like data set. If a model is not available,[12] a costly option is to create such a data set in the new language. A cheaper variant is to automatically translate the SQuAD data set into the new language, if MT systems are available. Another tractable option (Carrino et al., 2020) is to translate automatically the user's question into English, then align the answer found in the English Wikipedia page with the equivalent Wikipedia page from the other language, and return the aligned answer, which should thus be fluent and grammatically correct since it comes from an actual Wikipedia page.

[10] www.cs.cornell.edu/~cristian/Cornell_Movie-Dialogs_Corpus.html

[11] github.com/Phylliida/Dialogue-Datasets

[12] See a sample list at bertlang.unibocconi.it.

Acknowledgments

We thank the University of Applied Sciences of Western Switzerland (HES-SO) for the PLACAT grant that supported this work (AGP n. 82681).

6. Bibliographical References

Bunt, H., Alexandersson, J., Choe, J.-W., Fang, A. C., Hasida, K., Petukhova, V., Popescu-Belis, A., and Traum, D. (2012). ISO 24617-2: A semantically-based standard for dialogue annotation. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*, pages 430–437, Istanbul, Turkey. European Language Resources Association (ELRA).

Carrino, C. P., Costa-jussà, M. R., and Fonollosa, J. A. R. (2020). Automatic Spanish translation of the SQuAD dataset for multilingual question answering. In *Proceedings of the 12th International Conference on Language Resources and Evaluation (LREC-2020)*, Marseille, France. European Language Resources Association (ELRA).

Chen, D., Fisch, A., Weston, J., and Bordes, A. (2017). Reading Wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879, Vancouver, Canada. Association for Computational Linguistics.

Clark, A. and Popescu-Belis, A. (2004). Multi-level dialogue act tags. In *Proceedings of the 5th SIGdial Workshop on Discourse and Dialogue at HLT-NAACL 2004*, pages 163–170, Cambridge, MA, USA. Association for Computational Linguistics.

Devlin, J., Chang, M., Lee, K., and Toutanova, K. (2018). BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Dhingra, B., Li, L., Li, X., Gao, J., Chen, Y.-N., Ahmed, F., and Deng, L. (2017). Towards end-to-end reinforcement learning of dialogue agents for information access. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 484–495, Vancouver, Canada. Association for Computational Linguistics.

Fang, H., Cheng, H., Sap, M., Clark, E., Holtzman, A., Choi, Y., Smith, N. A., and Ostendorf, M. (2018). Sounding Board: A user-centric and content-driven social chatbot. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 96–100, New Orleans, Louisiana. Association for Computational Linguistics.

Harabagiu, S., Maiorano, S. J., and Paşca, M. (2003). Open-domain textual question answering techniques. *Natural Language Engineering*, 9(3):1–38.

Li, J., Monroe, W., Ritter, A., Jurafsky, D., Galley, M., and Gao, J. (2016). Deep reinforcement learning for dialogue generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202, Austin, Texas. Association for Computational Linguistics.

Popescu-Belis, A., Yazdani, M., Nanchen, A., and Garner, P. N. (2011). A speech-based just-in-time retrieval system using semantic search. In *Proceedings of the ACL-HLT 2011 System Demonstrations*, pages 80–85, Portland, Oregon. Association for Computational Linguistics.

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. (2016). SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Rajpurkar, P., Jia, R., and Liang, P. (2018). Know what you don't know: Unanswerable questions for SQuAD. *CoRR*, abs/1806.03822.

Rieser, V. and Lemon, O. (2009). Does this list contain what you were searching for? Learning adaptive dialogue strategies for interactive question answering. *Natural Language Engineering*, 15(1):55–72.

Rieser, V. and Lemon, O. (2011). *Reinforcement learning for adaptive dialogue systems: a data-driven methodology for dialogue management and natural language generation*. Springer-Verlag, Berlin/Heidelberg, Germany.

Stolcke, A., Ries, K., Coccaro, N., Shriberg, E., Bates, R., Jurafsky, D., Taylor, P., Martin, R., Ess-Dykema, C. V., and Meteer, M. (2000). Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational Linguistics*, 26(3):339–374.

Vinyals, O. and Le, Q. V. (2015). A neural conversational model. *CoRR*, abs/1506.05869.

Zhao, T. and Kawahara, T. (2017). Joint learning of dialog act segmentation and recognition in spoken dialog using neural networks. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 704–712, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Zhou, Y., Ziyu, X., Black, A. W., and Rudnicky, A. I. (2016). TickTock Re-Wochat 2016 shared task chatbot description report. In *Proceedings of the REWOCHAT Workshop at LREC 2016*, Portoroz, Slovenia. European Language Resources Association (ELRA).