Effective crowdsourced generation of training data for chatbots natural language understanding

Bapat, Rucha; Kucherbaev, Pavel; Bozzon, Alessandro

DOI
10.1007/978-3-319-91662-0_8

Publication date
2018

Document Version
Accepted author manuscript

Published in
Web Engineering - 18th International Conference, ICWE 2018, Proceedings

Citation (APA)
Bapat, R., Kucherbaev, P., & Bozzon, A. (2018). Effective crowdsourced generation of training data for chatbots natural language understanding. In Web Engineering - 18th International Conference, ICWE 2018, Proceedings (pp. 114-128). (Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics); Vol. 10845 LNCS). Springer. https://doi.org/10.1007/978-3-319-91662-0_8

Effective Crowdsourced Generation of Training Data for Chatbots Natural Language Understanding

Rucha Bapat, Pavel Kucherbaev, and Alessandro Bozzon

Delft University of Technology
Van Mourik Broekmanweg 6, Delft, 2628 CD, Netherlands

Abstract. Chatbots are text-based conversational agents. Natural Language Understanding (NLU) models are used to extract meaning and intention from user messages sent to chatbots. The user experience of chatbots largely depends on the performance of the NLU model, which itself largely depends on the initial dataset the model is trained with. The training data should cover the diversity of real user requests the chatbot will receive. Obtaining such data is a challenging task even for big corporations. We introduce a generic approach to generate training data with the help of crowd workers, we discuss the approach workflow and the design of crowdsourcing tasks assuring high quality. We evaluate the approach by running an experiment collecting data for 9 different intents. We use the collected training data to train a natural language understanding model. We analyse the performance of the model under different training set sizes for each intent. We provide recommendations on selecting an optimal confidence threshold for predicting intents, based on the cost model of incorrect and unknown predictions.

Keywords: Conversational Agents, Natural Language Understanding, Crowdsourcing

1 Introduction

Messenger applications, such as Facebook Messenger, Telegram, Whatsapp and WeChat, represent a popular medium of communication, which people use to interact with friends, colleagues, and companies. In 2015 the total number of active users of such applications surpassed the total number of users of conventional social network applications [12]. Chatbots, on the other hand, are computer programs living in messenger applications and emulating a conversation with a human to provide a certain service [19].

To make chatbots understand their users, natural language understanding (NLU) machine learning models process incoming messages and classify them according to a list of supported intentions – intent recognition – such as "get weather forecast" or "purchase a ticket", and identify associated information – entity or parameter extraction – such as the city for weather forecast or the destination for ticket purchase. In a live uncontrolled environment user phrasings are very diverse from lexical and syntactical perspectives. Users' messages might include grammatical mistakes, emojis, and ambiguous abbreviations. NLU is a crucial part of a chatbot, as if it fails, not matter how good other chatbot components (e.g. dialogue management, response generation) perform, the chatbot execution will likely be incorrect.

To ensure robust performance from the NLU model, its training should be performed upon a diverse high quality training data set, featuring a good coverage of messages the chabot will receive in production from live users. Acquiring such training set is not a an easy job, even for big companies. Such dataset should be labeled with intents and entities, and the number of available could training data greatly vary across domains.

While crowdsourcing is a suitable solution, the acquisition of high quality training data from open crowds is not trivial: brainstorming new request phrasings is a creative task, where quality control is harder to implement with respect, for instance, to more deterministic tasks like image labelling. While different approaches for collecting training data using crowdsourcing were introduced in the literature [14,2,16,20,21,28], we propose an end-to-end solution, which starts from an information need, a generic approach to collect relevant labelled examples, a collection of ways to enrich the training dataset, using this dataset to train an intent classifier, and a heuristic-based model suggesting an optimal confidence threshold for this classifier in order to achieve business goals set for the conversational agent. In this paper, we provide the following original contributions:

- A *domain-independent end-to-end approach* to generate high quality training data for chatbot's NLU using the crowd, enriching this data, and training intent classifier;
- A *data collection experiment* where we collect training data using a crowdsourcing platform for 9 different intentions from 3 diverse domains and evaluate its quality;
- An *NLU model training experiment* where we train a model with the collected data and evaluate its performance with training sets of a different size;
- An *approach* that supports the selection of an optimal confidence threshold for the intent classifier, by means of a cost function that accounts of the costs caused by incorrect and unknown classifications.

The remainder of the paper is organised as follows: Section 2 presents the end-to-end pipeline that constitutes our domain independent approach. Section 3 details the data generation step, and presents and discusses the results of a data collection experiment. Section 4 studies the performance of an NLU model trained with the content generated in the previous step, and discusses a cost function for the selection of an optimal confidence threshold. Section 5 presents related work. Section 6 concludes.

2 End-to-end pipeline

Our pipeline is designed to target chatbots serving information retrieval (e.g. "when is the next train to the airport?") or transactional (e.g. "purchase a ticket for the next train.") purposes, rather than ones aiming to sustain a generic conversation with users. As such, the chatbot could be seen as a user interface for a Web service.

Lets assume the chatbot's Web service for which we plan to generate training set is REST API. Then API endpoints map with message intents, and API endpoint parameters map with entities. Figure 1 depicts the pipeline, which consists from 3 main stages:

1. Dataset Generation
2. Dataset Enrichment
3. Model Training

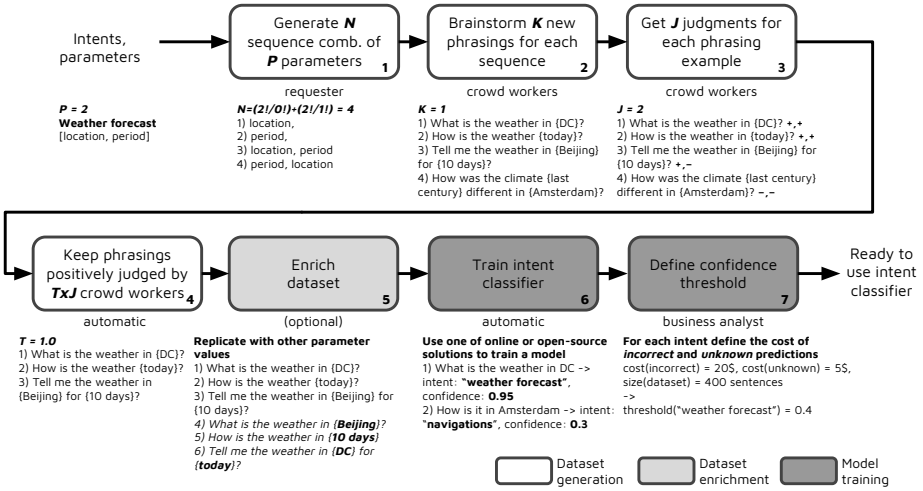

Fig. 1. Dataset generation pipeline for training chatbots natural language understanding models.

The goal of the first stage is to collect request examples from the crowd for given intents. The goal of the second stage is to enlarge the dataset collected in stage i) using various techniques. In the last stage we train an intent classifier and tune it to meet business needs of the conversational agent being developed.

Below we extensively discuss stage 1 (Section 3) and stage 3 (Section 4). We skip the enrichment stage due to space limit, which and refer the interested reader to existing literature [14,21].

3 Dataset Generation

Let us assume the development of a chatbot to get weather forecast using some API. For simplicity reasons, we let users get a forecast by two parameters: location and period. Users of the chatbot can ask for a forecast with using different phrasing and mentioning these parameters (not necessarily all of them) in any order. Valid request examples would be: "How is it in *San Francisco* today?", "What is the weather *next week*?" (location is missing).

As a result of extensive experiments with crowd workers, we consolidated the following 5 steps approach to generate a dataset to train NLU model (Figure 1):

1. Generate N sequence combinations of P parameters. In the real world, users could submit requests to the chatbot mentioning parameters in any order. Having this assumption in mind, a reasonable strategy for training set generation would be to collect several request examples for every possible sequence of parameters. For that we create a list of possible parameter sequences, which we later use in the brainstorming task.

The number of possible sequences is calculated as:

$$Nsequences = \sum_{i=0}^{P-1} \frac{P!}{i!}$$

, where P is the number of parameters.

2. Brainstorm K new phrasings for each sequence. For each sequence from the pool of sequence combinations we collect K examples from the crowd. As in live scenario chatbot users will write their requests with high diversity of phrasing, we need to account for such diversity in this step. The fact that we ask people from different countries and with different background and demographics already injects some diversity in the examples. To ensure the creation of reasonable amount of examples from the crowd, we introduce the following quality control techniques (including validators as in Figure 2).

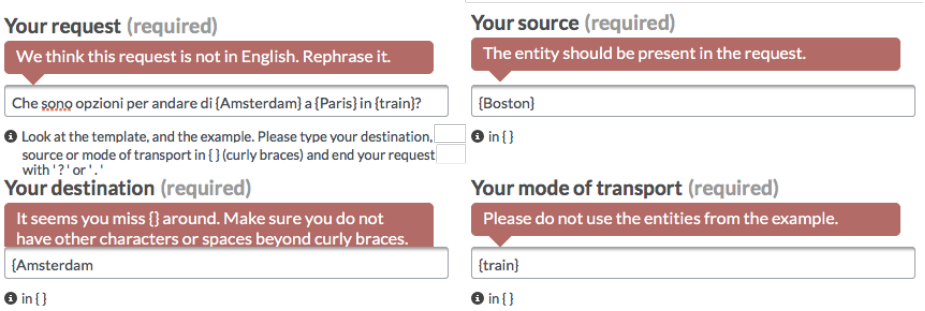

Fig. 2. An example of how validators work in the brainstorming task on CrowdFlower.

Instructions - a short instruction is given to workers, where we describe the expect contribution;

Example - to give a feeling about the expected result we give an example of a request example which fits the given sequence;

Language control - even though we ask workers to provide requests in a given language (e.g. English), some workers might provide examples in other languages. To control this, we use a third party service (`https://detectlanguage.com`) which we call using JavaScript before the task form is submitted, and if the service predicts language different than English we give feedback to the worker and ask to write it in English instead;

Sequence control - even though we ask workers to follow the requested sequence of parameters, we need to assure it, so we validate the sequence before the task form is submitted;

Uniqueness control - even thought we ask workers to come up with their own phrasing and parameter values, some where copying our example and were adding a letter in the end of the sentence. To address that we calculate *Levenshtein* distance of our example and the one by the worker and see if they are far apart enough.

3. Get Judgments for each phrasing example Despite the use of different validators. it is still possible to obtain low quality requests from crowd workers. To address this problem we introduce another layer of quality control, where we launch another crowdsourcing task and ask several workers to evaluate requests we collected from the brainstorming task. To make sure workers understand well this validation task we have some ground truth data and give real tasks only to people who pass the qualification round of the task.

4. Keep phrasings positively judged by $T \times J$ crowd workers For each request from the brainstorming task we have several judgments about its fitness to the specified intent. Depending on the specified requirement, we define some agreement threshold, which is used to use this request to train NLU model or not:

$$A = \left.\begin{cases} 1, & \text{if } \frac{J_{positive}}{J_{positive}+J_{negative}} \geqslant T_{accept} \\ 0, & \text{if } \frac{J_{positive}}{J_{positive}+J_{negative}} < T_{accept} \end{cases}\right\}, T_{acceptance} \in [0,1]$$

5. Enrich phrasings More training examples could be automatically generated by: i) sentence paraphrasing [14,21], ii) adding some extensions (e.g. "what is the weather today?" $\rightarrow$ "*hey chatbot,* what is the weather today?"), iii) replacing parameter values with others (e.g. "what is the weather *today*?" $\rightarrow$ "what is the weather *tomorrow*?"), iv) generating big pools of parameter values from open data sources (e.g. thousands of location options could be generated from Google Maps API), v) swapping two or more consecutive letters to account for possible misspellings (e.g. "what is the w*ae*ther today?"). These and many other strategies can help to increase the size of the training set and make the NLU model more robust.

3.1 Experiment setup

To test the effectiveness of our approach we conduct an experiment, where we collect data for 9 intents (Table 1) in 3 different domains: travel information (which is popular among chatbots selling tickets and providing timetable), meeting scheduling (multiple chatbots, such as X.ai and Calendar.help [6], work in the same domain), and software development (to challenge our approach, as it is generally perceived that it is only possible to perform tasks on crowdsourcing platforms not requiring a specific knowledge). In each domain we test the approach with 3 types of intents: *1) read* - where users retrieve some information, *2) create* - where users intend to perform a new transaction, *3) update* - where users intent to edit information.For intents with 3 parameters (15 sequence combinations) we requested examples from 7 workers, for intents with 4 parameters (64 sequence combinations) – from 4 workers. Later each request is judged by 3 workers in the validation task.

As the main focus of this work is on getting high quality results from crowd workers, we test only the first 4 steps of our approach, considering that the data enrichment step is a topic on its own, requiring a separate extensive analysis. We launch all tasks on CrowdFlower not concurrently, at different working days at the same time span. To make sure the results we collect are representative we repeat all tasks 3 times.

Table 1. We collect training data for 9 intents of 3 types (read, create, update) from 3 domains (travel information, scheduling meetings, software development).

		TRAVEL	MEETING	SOFTWARE
READ	Intent	Ask for navigation	Availability check	"How to" questions
	Parameters	source, destination, mode of transport	time, alternative time, place	progr. language, OS, package/tool
CREATE	Intent	Purchase a ticket	Create a meeting	Deploy software
	Parameters	source, destination, trip purpose, date	time, participants, place, duration	action, OS, memory requirement
UPDATE	Intent	Modify a ticket	Modify a meeting	Modify software
	Parameters	source, destination, date	time, participants, place, duration	error, progr. language, OS, package/tool

3.2 Metrics

We first manually check all the results coming from the brainstorming step of the pipeline. To evaluate the effectiveness of our approach we use the following formulas to calculate accuracy before validation step (A_{bv}), accuracy after the validation step (A_{av}):

$$A_{bv} = \frac{N_{correct}}{N_{correct} + N_{incorrect}}, \quad A_{av} = \frac{N_{correct \cap accepted}}{N_{accepted}}$$

3.3 Results

The results[1] of the experiment are summarized in Table 2. The mean accuracy of the brainstorming task before the validation task is 88.66% (standard deviation = 5.72). Varying the threshold of acceptance T_{accept} we got the following accuracies: 93.57% (for $T_{accept} = 0.33$), 96.37% (for $T_{accept} = 0.33$), and 98.21% (for $T_{accept} = 1$). The higher the threshold of acceptance, the higher the mean accuracy and lower its standard deviation. The lowest accuracy (86.53%) is for modify ticket in travel domain with acceptance threshold = 0.33. The highest accuracy (99.85%) is for creating meeting intent with acceptance threshold = 1. From the diversity perspective 99.8% of collected examples are different. If we fill up parameters (slots) with the same values, then 77% of requests are different (meaning that the rest have the same phrasing but different parameter values). Here are some request examples collected from the crowd:

- Valid example: *What is the best route to go by {car} to {CN Tower} from {Yonge Station}*
- Invalid example, caught in review task: *go to {vegas} to {boise}*
- Invalid example, not caught in review task: *How to go to {public transport} from {Dasmarinas City}?*

[1] The sequences of parameters for each intent, examples collected in brainstorming task, and judgments from the validation task are available here https://github.com/HumanAidedBots/NLU

Table 2. Training data generation pipeline results. All results are averaged over 3 repetitions.

		BRAINSTORMING	VALIDATION		
		Accuracy, %	Accuracy, % T = 0.33	Accuracy, % T = 0.66	Accuracy, % T = 1.00
TRAVEL	Read	90,15	97,57	98,92	99,58
	Create	94,39	97,14	98,7	99,63
	Update	79,36	86,53	90,31	94,87
MEETING	Read	88,25	91,47	96,46	97,78
	Create	98,82	99,47	99,73	99,85
	Update	81,85	92,46	95,87	99,2
SOFTWARE	Read	89,83	93,71	97,09	98,21
	Create	90,79	94,17	96,49	97,16
	Update	84,5	89,59	93,72	97,63
	Mean	**88,66**	**93,57**	**96,37**	**98,21**

3.4 Discussion

Crowd workers brainstormed examples for read and create intents very well, but shown problems with update intents. Further analysis shown that crowd workers considered update intent tasks more confusing, which also explains the poor performance in the validation step. This suggests the need for improved user interface and instructions for such tasks, to enable the generation of better content.

The fact that we achieve 97% accuracy for read and create intents, suggests that the method can already be used for acquiring large corpus of training data, as such level of accuracy was achieved for all 3 domains.

For parameters most of the workers provided very short examples (e.g. not full addresses, company names). Several request examples contain grammar mistakes (e.g. "How to go from faro by *plain*?" all workers in the validation task accepted, while "*w would* I go from Lima by train to Ica?" all workers rejected). This is an interesting observation, that shows how crowd workers are likely to have mistakes and typos while interacting with a chatbot as well.

In our work we do not focus a lot on enforcing diversity, apart from the fact that we collect examples from a diverse group of people from different countries and we double check that they provide examples which are significantly different (using Levenshtein distance) than the example we give them.

There are multiple methods could be applied to assure diversity (e.g. GWAP style game where workers need to come up with phrasings never used by other workers, but at the same time rated as highly relevant). We discuss such approach as future work.

4 NLU Model

To provide a measure of the quality of the collected sentences, we study how the performance of NLU model trained with the content generated in the data collection experiment varies for different training data sizes and different intents. It is often not reasonable to blindly trust predictions given by the model. Therefore, we as well study how the NLU model prediction combined with a parametric threshold level can be used to trade off between incorrect and unknown predictions (those which have confidences below the threshold).

We acknowledge that testing the performance of the NLU model trained with data generated using the same approach can lead to over fitting, and that an evaluation performed with test data coming from a different source (ideally from a live chatbot with real users) would be more appropriate. As we lack access to such data, the goal of our experiment is to explore how the performance of the trained NLU model vary with alternative training data sizes and confidence thresholds.

4.1 Experiment setup

In our experiments we used the popular open-source platform Rasa NLU[2]. This platform allows to train intent classifier and use this classifier *locally* with no need to have internet connection and send data to third-party services. Rasa NLU intent classifier is based on support vector machines (SVM). Further we only discuss the performance of the intent classifier (not the entity recognition) with a training data collected in the previous section.

We divide 4158 request examples we collected from the crowd into 90% (3742) training set and 10% (416) test set. We vary training size from 15 to 3742 request examples (approximately doubling it every time, so we have 10 possible sizes). We keep our training sets and the test set balanced with respect to the number of requests from each intent. We performed an N-fold validation, where having multiple training sets for each training size (e.g. 5 sets, for 15, 30, 59, 117, 234, and 468; 3 sets 936, and 1 set for 2700 and 3742 dataset sizes) are used for training, and the resulting performance averaged.

4.2 Results

Figure 3 reports the performance of NLU model trained with different dataset sizes. For each setting we show 9 stacked bars corresponding to the number of correct predictions (green), incorrect predictions (red) and unknowns (grey) - predictions with prediction confidence less than the confidence threshold (which we vary from 0.1 to 0.9 with a step of 0.1). The performance for different intents is not consistent, such with a very small training data set of just 30 training examples the performance of Travel Read is over 90%, while Software Create is below 10%. For intent Meeting Create with training size 3742 all test requests were classified correctly.

[2] https://rasa.ai

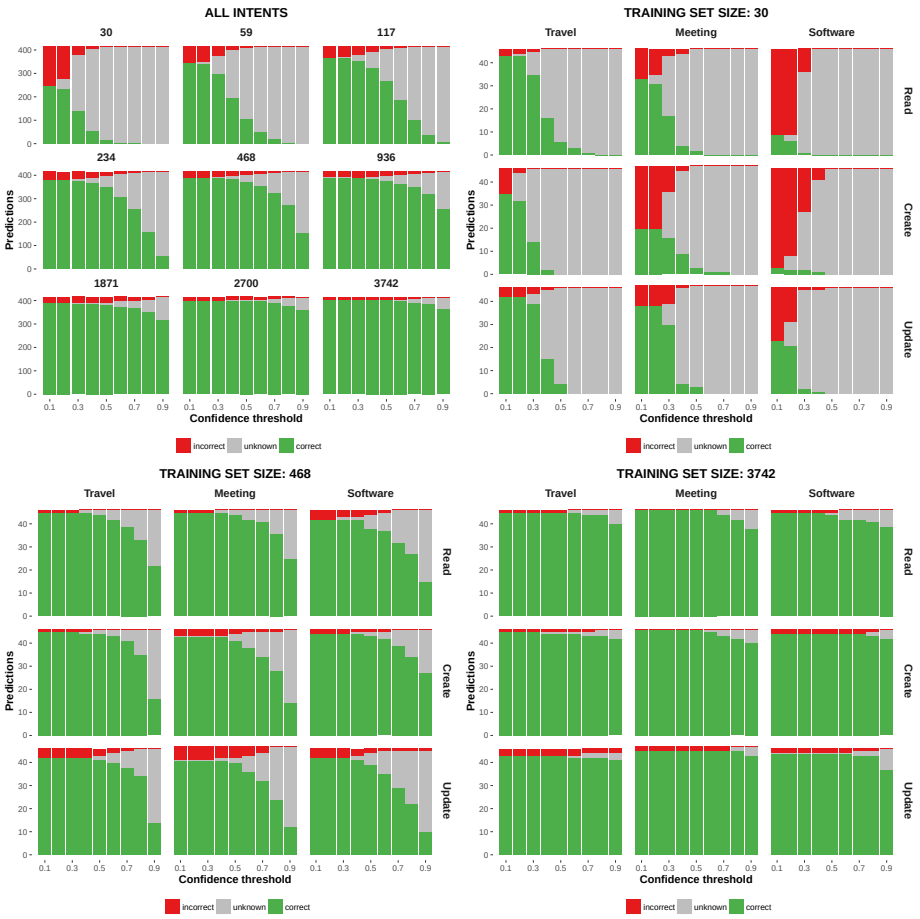

Fig. 3. Performance comparison of NLU models trained with data of a different size. In the top-left – combined performance across all intents, other 3 quadrants – detailed results for each intent for a given training set size.

Looking at the combined performance (the bottom part of the figure) with the training data of less than 117 request examples the model has very low performance (less than 90%). Increasing the training data 16 times helps to reach 95% accuracy level. Further dataset size increase gives only slow improvement in its performance. Interesting to notice, that the model with training set 117 and confidence threshold 0.5 gives the same number of incorrect predictions as the model with training set 3742 and confidence threshold 0.1.

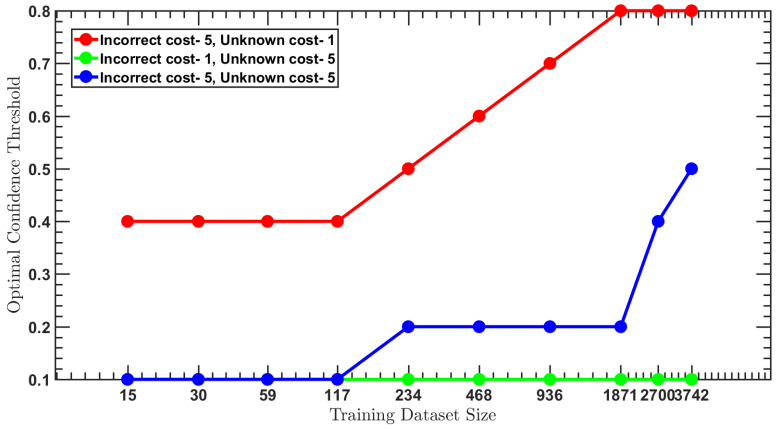

Fig. 4. Optimal thresholds for different training sizes for 3 different cost models (red – incorrect classifications are more expensive, blue – incorrect and unknown classifications have the same cost, green – unknown classifications are more expensive).

4.3 Discussion

Different intents assume different level of diversity of user requests, such questions about navigation information (Travel Read) or purchasing a ticket (Travel Create) are quite standard and typical, while questions about meeting scheduling or software development vary greatly, therefore requiring more training examples to reach reasonable performance. Still with training sets of several thousands (e.g. 3742) the performance across different intents is quite consistent.

4.4 Define Confidence Threshold

Changing the confidence threshold leads to different number of incorrect and unknown classifications (Figure 3). Conversational agents are often built to solve business tasks, such as selling airplane tickets or hotel reservations. To pick the optimal confidence threshold the system designer needs to identify a potential cost of incorrect classification and a cost of unknown classification. This cost can be defined in time, money, or a number of interactions users have to go through to reach their goal. Further we give an example, how to come up with a cost model and what is the optimal confidence threshold in different cost models.

Let's say that it is possible to earn 10 USD with each flight ticket reservation. Looking at hypothetical historical data, we might say that an incorrect intent classification drops the probability of the user to make reservation in half. Such, the cost of incorrect classification is 5 USD. Unknown intent classifications lead to extra clarification questions, which lead to the drop of the booking probability by 10%. Such, the cost of unknown classification is 1 USD.

In Figure 4 we show optimal confidence thresholds for 3 different cost models: i) incorrect – 5 USD, unknown – 1 USD, ii) incorrect – 1 USD, unknown – 5 USD, iii)

incorrect – 5 USD, unknown – 5 USD. We came up with this figure based on the data given in the top-left quadrant of Figure 3, using the following cost function:

$$TotalCost = N_{incorrect} \times C_{incorrect} + N_{unknown} \times C_{unknown}$$

, where N is the number of classifications, and C is the cost of correct or incorrect classifications. Such, if unknowns are more expensive (green line), then it makes sense to define the confidence threshold as very small, introducing some incorrects, because unknowns are anyway more expensive. If the cost of incorrects and unknowns are the same (blue), then we just need to minimize their total number, maximizing the number of correct classifications. The most common case, is when incorrect classification is more expensive than an unknown one (red). In this case with smaller training sets the optimal threshold is 0.4, and as we grow the training set the optimal threshold grows towards 0.8.

5 Related Work

Crowdsourcing and human computation are widely used to allow conversational agents and chatbots to understand their users. *Chorus* is a conversational agent, where all users requests are processed by multiple crowdworkers [18,11]. *Guardian* is a conversational agent where crowd workers ask questions to the users to derive parameters and their values to run a request with an underlying API [8]. In *InstructableCrowd*, users converse with crowd workers to create if-then rules for their smartphones to perform various actions [9]. While in all previous systems every user request was processed solely by crowdworkers, the *CRQA* question and answering system make automatic algorithm and crowd workers work in parallel to suggest answers to user requests, with crowd workers voting for the best answer [23]. Huang et. al. introduce an approach to perform real-time entity extraction with the crowd in chatbot systems [10]. Vtyurina et. al. compare satisfaction and human behaviour of users interacting with a human expert and a system perceived to be automatic, but backed by a human worker [27].

While there are many examples of conversational agents where the crowd is involved in processing every request this is not scalable (at least financially). In *Calendar.help*, an email-based conversational agent helping people to arrange meetings, multiple tiers aim to understand text of emails: automatic natural language understanding model, crowd micro-tasks (e.g. to identify meeting location), and crowd macro-tasks (e.g. in case other tiers fail, to understand how to proceed with the email and write another email to reply with) [6]. While such approach seems to be more scalable, the natural language understanding model requires a dataset to be trained upon.

The importance of rich and diverse dataset for training dialogue systems is discussed in [25]. There is some research has been done about using the crowd to collect training data for conversational agents. Most of this research address spoken language collection (e.g. via Amazon Mechanical TURK [15]), and transcription and annotation of speech corpora [7]. Rothwell et. al. discuss collection and annotation of named entity recognition data using unmanaged crowds [22].

Human computation was historically used a lot for processing or generating textual information, be it Soylent Word plugin for improving text and proofreading [1], image

caption generation [3] using computer games [26], text simplification [17], translation quality evaluation [5], and evaluation of other natural language tasks [24]. Jha et. al. introduce an approach to curate prepositional phrase attachment corpus by having automated point prediction system and crowd workers from MTURK working together [13].

Crowdsourcing could be considered as a way to help expert annotators to come up with training examples for dialogue systems [20]. Lasecki et. al. [16] propose to acquire dialogue training data by crowdsourcing conversations between pairs of crowdworkers towards defined goals. In [28] the authors propose several methods using crowdsourcing to collect training sentences matching given semantic forms. Sentence paraphrasing using crowdsourcing [14,21] is a way to increase training dataset.

6 Conclusion and Future Work

We introduced an end-to-end pipeline to generate training data for conversational agents and training natural language understanding models. In this pipeline we introduced new approaches, such as validation techniques in task user interface, but also refer to concepts from the literature (e.g. going over combinations of intents and parameters, and using paraphrasing for data enrichment).

We conducted an experiment collecting training data for 9 different intents from 3 different domains and on average with the most strict validation policy we collected request examples with 98.21% accuracy. Later we trained NLU model with the data we have collected using our approach and reported how the performance differ with training set size and intent. In addition we introduced various cost models for incorrect and unknown classifications, and reported how different confidence thresholds could be used to meet business goals.

In the current work we evaluated the performance of the NLU model splitting the dataset collected from the crowd into training and testing sets. Even thought it provides some intuition on the performance of the model in real life scenario, the first thing we plan to do in the future is to test the model with requests collected from an online chatbot running with real users. In addition we plan to improve the diversity of requests we generate with the approach, by allowing crowd workers to submit only new unique phrasings for a given intent (similar to image tagging in games with a purpose [4]). We plan to investigate different techniques of phrasing enrichment (e.g. auto-generating parameter values, adding extra noisy attachments to requests) to improve the performance of the NLU model.

Acknowledgments This research has been supported in part by the Amsterdam Institute for Advanced Metropolitan Solutions with the AMS *Social Bot* grant, and by the Dutch national e-infrastructure with the support of SURF Cooperative (grant *e-infra170237*).

References

1. M. S. Bernstein, G. Little, R. C. Miller, B. Hartmann, M. S. Ackerman, D. R. Karger, D. Crowell, and K. Panovich. Soylent: A word processor with a crowd inside. In *Proceedings of the 23Nd Annual ACM Symposium on User Interface Software and Technology*, UIST '10, pages 313–322, New York, NY, USA, 2010. ACM.
2. A. Bozzon, M. Brambilla, S. Ceri, A. Mauri, and R. Volonterio. Pattern-based specification of crowdsourcing applications. In S. Casteleyn, G. Rossi, and M. Winckler, editors, *Web Engineering*, pages 218–235, Cham, 2014. Springer International Publishing.
3. A. Bozzon, I. Catallo, E. Ciceri, P. Fraternali, D. Martinenghi, and M. Tagliasacchi. A framework for crowdsourced multimedia processing and querying. In *Proceedings of the First International Workshop on Crowdsourcing Web Search, Lyon, France, April 17, 2012*, pages 42–47, 2012.
4. A. Bozzon and L. Galli. *An Introduction to Human Computation and Games with a Purpose*, pages 514–517. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.
5. C. Callison-Burch. Fast, cheap, and creative: Evaluating translation quality using amazon's mechanical turk. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1 - Volume 1*, EMNLP '09, pages 286–295, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.
6. J. Cranshaw, E. Elwany, T. Newman, R. Kocielnik, B. Yu, S. Soni, J. Teevan, and A. Monroy-Hernández. Calendar.help: Designing a workflow-based scheduling agent with humans in the loop. January 2017.
7. J. Dias. Transcribing and annotating speech corpora for speech recognition: A three-step crowdsourcing approach with quality control. In *1st AAAI Conference on Human Computation and Crowdsourcing, HCOMP 2013*, volume WS-13-18, pages 30–31, Palm Springs, 2013. Association for the Advancement of Artificial Intelligence., AI Access Foundation.
8. T. H. Huang, W. S. Lasecki, and J. Bigham. Guardian: A crowd-powered spoken dialog system for web apis. In *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, HCOMP '15. AAAI, 2015.
9. T.-H. K. Huang, A. Azaria, and J. P. Bigham. Instructablecrowd: Creating if-then rules via conversations with the crowd. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, CHI EA '16, pages 1555–1562, New York, NY, USA, 2016. ACM.
10. T. K. Huang, Y. Chen, and J. P. Bigham. Real-time on-demand crowd-powered entity extraction. *Collective Ingelligence*, 2017.
11. T. K. Huang, W. S. Lasecki, A. Azaria, and J. P. Bigham. "is there anything else I can help you with?": Challenges in deploying an on-demand crowd-powered conversational agent. In *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, HCOMP '16, 2016.
12. B. Insider. *Messaging apps are now bigger than social networks*, sep 2015. http://www.businessinsider.com/the-messaging-app-report-2015-11.
13. M. Jha, J. Andreas, K. Thadani, S. Rosenthal, and K. McKeown. Corpus creation for new genres: A crowdsourced approach to pp attachment. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, CSLDAMT '10, pages 13–20, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.
14. Y. Jiang, J. K. Kummerfeld, and W. S. Lasecki. Understanding Task Design Trade-offs in Crowdsourced Paraphrase Collection. *ArXiv e-prints*, Apr. 2017.
15. I. Lane, A. Waibel, M. Eck, and K. Rottmann. Tools for collecting speech corpora via mechanical-turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech*

and Language Data with Amazon's Mechanical Turk, CSLDAMT '10, pages 184–187, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.
16. W. Lasecki, E. Kamar, and D. Bohus. Conversations in the crowd: Collecting data for task-oriented dialog learning. In *Scaling Speech, Language Understanding and Dialogue through Crowdsourcing Workshop*. AAAI, January 2013.
17. W. S. Lasecki, L. Rello, and J. P. Bigham. Measuring text simplification with the crowd. In *Proceedings of the 12th Web for All Conference*, W4A '15, pages 4:1–4:9, New York, NY, USA, 2015. ACM.
18. W. S. Lasecki, R. Wesley, J. Nichols, A. Kulkarni, J. F. Allen, and J. P. Bigham. Chorus: A crowd-powered conversational assistant. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology*, UIST '13, pages 151–162, New York, NY, USA, 2013. ACM.
19. M. McTear, Z. Callejas, and D. Griol. *The Conversational Interface: Talking to Smart Devices*. Springer, Cham, 2016.
20. M. Negri, L. Bentivogli, Y. Mehdad, D. Giampiccolo, and A. Marchetti. Divide and conquer: Crowdsourcing the creation of cross-lingual textual entailment corpora. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 670–679, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.
21. M. Negri, Y. Mehdad, A. Marchetti, D. Giampiccolo, and L. Bentivogli. Chinese whispers: Cooperative paraphrase acquisition. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation, LREC 2012, Istanbul, Turkey, May 23-25, 2012*, pages 2659–2665, 2012.
22. S. Rothwell, S. Carter, A. Elshenawy, V. Dovgalecs, S. Saleem, D. Braga, and B. Kennewick. Data collection and annotation for state-of-the-art NER using unmanaged crowds. In *INTERSPEECH 2015*, pages 2789–2793, 2015.
23. D. Savenkov and E. Agichtein. Crqa: Crowd-powered real-time automatic question answering system. In *Fourth AAAI Conference on Human Computation and Crowdsourcing*, HCOMP '16, 2016.
24. R. Snow, B. O'Connor, D. Jurafsky, and A. Y. Ng. Cheap and fast—but is it good?: Evaluating non-expert annotations for natural language tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 254–263, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics.
25. G. Tür, D. Hakkani-Tur, and L. P. Heck. What is left to be understood in atis? In D. Hakkani-Tür and M. Ostendorf, editors, *SLT*, pages 19–24. IEEE, 2010.
26. L. von Ahn and L. Dabbish. Labeling images with a computer game. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '04, pages 319–326, New York, NY, USA, 2004. ACM.
27. A. Vtyurina, D. Savenkov, E. Agichtein, and C. L. A. Clarke. Exploring conversational search with humans, assistants, and wizards. In *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, CHI EA '17, pages 2187–2193, New York, NY, USA, 2017. ACM.
28. W. Y. Wang, D. Bohus, E. Kamar, and E. Horvitz. Crowdsourcing the acquisition of natural language corpora: Methods and observations. In *2012 IEEE Spoken Language Technology Workshop (SLT), Miami, FL, USA, December 2-5, 2012*, pages 73–78, 2012.