Assessing the Robustness of Conversational Agents using Paraphrases

Jonathan Guichard*, Elayne Ruane*, Ross Smith†, Dan Bean†, Anthony Ventresque*
*School of Computer Science, University College Dublin, Ireland &
Lero - the Irish Software Research Centre
Email: elayne.ruane@ucdconnect.ie, anthony.ventresque@ucd.ie
†Microsoft Corporation, Skype Division, Seattle, USA.

***Abstract*—Assessing a conversational agent's understanding capabilities is critical, as poor user interactions could seal the agent's fate at the very beginning of its lifecycle with users abandoning the system. In this paper we explore the use of paraphrases as a testing tool for conversational agents. Paraphrases, which are different ways of expressing the same intent, are generated based on known working input by performing lexical substitutions. As the expected outcome for this newly generated data is known, we can use it to assess the agent's robustness to language variation and detect potential understanding weaknesses. As demonstrated by a case study, we obtain encouraging results as it appears that this approach can help anticipate potential understanding shortcomings and that these shortcomings can be addressed by the generated paraphrases.**

1. Introduction

A *Conversational Agent* is a piece of software intended to dialogue with a human user to provide specific services, serve as a virtual assistant, or take part in social conversations. In recent years, these agents, also known as *chatbots*, have become increasingly popular. Many companies see them as a cost effective solution to handle great volumes of simple customer requests while users enjoy a convenient, fast, and always accessible service. Big "tech" firms have recently developed their own personal assistants, such as Apple's Siri or Microsoft's Cortana, while both Google and Amazon released physical devices based on their assistant.

Despite the recent progress made in the field, conversational agents still seem to struggle to win over peoples' hearts: according to some analysts, their current retention rate remains low, at an average of just 4% on a 7-day time frame [6]. As with any service or product, the final quality of these agents is assessed by the user. Characteristics such as efficiency, reliability, and effectiveness will impact their willingness to continue to use the service. When it comes to conversational agents, accurately interpreting user input is a key factor in the final user experience and thus in user perception of agent quality. An agent unable to understand the natural language of its users, forcing them to repeatedly rephrase their requests, is unlikely to have a high retention rate. Therefore, the ability to assess the agent's understanding of user input during the development phase could provide useful insight and ultimately lead to an improvement in user experience.

In this paper, we investigate how to automatically test the robustness of text-based conversational agents to language variation, limiting our study to the intent recognition aspect of those agents. In the context of Natural Language Understanding, intent recognition consists of labelling a user query according to the perceived nature of this query. For example, queries such as "i need to cancel my ticket" or "why cant i see my contacts" could lead an agent to classify those as "CancellationRequest" and "ContactsListHelp" requests respectively. In the context of Linguistics, language variation refers to the different ways of expressing the same semantic content. Variations can take the form of a different word choice or a different syntax, and are often related to the regional and socio-economic background of the speaker.

We only consider agents that can be represented as an automaton, with the transitions being queries formulated by the user and each state triggering an action and a written response from the agent. This structure makes it easy to verify that an agent correctly understands a query by checking the resulting state. This therefore excludes agents with a much more complex structure, for example agents that try to have social conversations with their users, such as *cleverbot*[1].

Briefly put, the problem can be summarised as automatically performing *divergent example testing* for conversational agents [19]. Divergent examples are similar to adversarial examples as defined in [11] as "a sample of input data which has been modified [...] in a way that is intended to cause a machine learning classifier to misclassify it". In our case, we wish to generate divergent examples that retain the same meaning while expressing it in a range of different ways: we call this operation *paraphrasing*.

The research question we are examining is "Can paraphrasing techniques be used to evaluate the understanding capabilities of Conversational Agents?" – and our contribution is an evaluation of the work we presented in Ruane et al. [19] that described a testing framework for conversational agents. In particular, we show that divergents, such as our paraphrases, can be used to discover weaknesses of a conversational agent. We also address the related research

1. http://www.cleverbot.com/

questions "Can paraphrases be used to enrich the training models of conversational agents?" – the decision being between giving more (independent) data to the language models of the conversational agents (more examples) or generating and using paraphrases.

The remainder of this paper is organised as follows: Section 2 presents a literature review of the paraphrase generation process and discusses work related to divergent testing; Section 3 gives a description of our testing framework; Section 4 presents our paraphrase generation module; Section 5 is an evaluation of our technique to test conversational agents; and Section 6 concludes our study.

2. Related Work

The process of automatically paraphrasing a given input sentence can be broken down in to two main steps; the acquisition of paraphrasing data and rules, and the generation of new sentences using this knowledge. According to Barzilay and McKeown [1], there are three major approaches to the problem: manual collection of paraphrases, utilization of lexical resources, and corpora-based acquisition.

The manual collection of paraphrases is usually performed with the use of crowd-sourcing platforms, such as Amazon's Mechanical Turk. This type of approach is notably used by Chklovski [5], where the corpus of paraphrases is built through a game in which users are given a sentence that they must reformulate, the goal being to correctly guess a hidden reference paraphrase. The guesses entered by the users during this game can then be used as new paraphrases. Other tasks that have been successfully used for manual collection of paraphrases include describing a clear and unambiguous action performed in a very short video clip [2], and asking users to directly select sentences that are paraphrases of each other with the sentences being extracted from tweets of trending topics [23].

The most notable work based on lexical resources has been proposed by Hassan et al. [10]. The authors propose a solution to the SemEval 2007 Lexical Substitution task which consists of finding the most probable synonym of an ambiguous word given its context. Their solution involves multiple scoring methods such as language models, word-sense disambiguation, and latent semantic analysis, amongst others. The authors claim to consistently rank in first or second place for this task.

The corpora based approaches to paraphrasing are the most commonly encountered in the literature. They aim to collect paraphrases from texts which are close in meaning, such as news articles discussing the same events or multiple translations of the same book. The approach proposed by Shinyama et al. [20] is one of the first to extract paraphrases from text corpora by leveraging different translations of the same news articles. Paraphrases are identified based on common named entities (such as proper nouns), and are then used to build a synchronous context-free grammar which is used for the generation process. This method is able to achieve interesting but low-coverage results. Similarly to Machine Translation, Quirk et al. [17] approach paraphrasing by considering it as a statistical problem. The paraphrases are also learned based on news articles using statistical machine translation tools, but the generation process consists of finding the optimal path through a graph representing the multiple rephrasing possibilities and their associated likelihood given the input sentence. Zhao et al. [24] extend the method by combining multiple resources in order to improve coverage. While paraphrases of sentences from news articles achieve a human acceptance rate of up to 70%, informal sentences from forums are only deemed acceptable 40% of the time.

Paraphrasing with a target style has also been investigated in previous work, as done by Xu [23] who proposed a Shakespeare/Modern English "translation" tool. The approach is based on the method proposed by Quirk et al. [17] using translations of Shakespeare's plays into Modern English as the input data. The same method is also applied with "Internet English" as the target style by using Twitter as the data source.

Some efforts have also been made to build databases of known paraphrases, as done by Callison-Burch et al. [8], [16]. In their papers, the authors perform paraphrase acquisition through a "bilingual pivoting method". The core idea of their approach is that if two English phrases translate to the same foreign phrase, they can be considered paraphrases of each other. The second version of their paraphrasing database for English contains more than 100 million entries which are expressed in the form of synchronous context-free grammar rules such as: $\langle$ *the* X_1 *of* X_2, X_2 *'s* X_1 $\rangle$. These paraphrasing rules have been further used by Napoles et al. [15] in their ready-to-use paraphrasing engine based on the statistical machine translation engine *Joshua*. The authors claim that their solution is the first one to provide researchers with a ready-to-use paraphrasing engine.

As with many other fields, deep learning is now being applied to paraphrasing, as done by Gupta et al. [9]. The approach they propose is a combination of generative models (based on Variational Auto-Encoders) with sequence-to-sequence models (based on Long Short-Term Memories), and is able to generate a paraphrase for a given input sentence. The training data is made up of original sentences and their reference paraphrase. According to the authors, their technique outperforms the state-of-the-art by a significant margin and sets a new baseline for future research.

As mentioned in Section 1, divergent example testing [19] is conceptually similar to the use of adversarial examples to test machine learning systems [11]. However, the goal of divergent testing is not to trick the conversational agent *per se*, rather, we try to mimic and emulate the complexity that is embedded in the way users naturally express themselves. This entails phenomenons such as the use of slang words and colloquial expressions (in the case of paraphrase generation), as well as grammar, spelling, and homophone mistakes. Similarly, divergent testing is related to metamorphic testing as proposed by Chen et al. [3] in 1998. Metamorphic Testing (MT) leverages relations between expected output of multiple related inputs and has been successfully applied in numerous domains. [4] Like

MT, divergent testing utilises working input (successful test cases) but does not require a formal process of identifying "metamorphic relations".

Finally, and to the best of our knowledge, we couldn't find any other previous work regarding the use of paraphrasing techniques as a testing tool for conversational agents.

3. Divergent Testing Framework

As presented previously, the context in which this work takes place is the testing of conversational agents, based on the principle of *divergent example testing* [19]: a base-case is altered while retaining the same expected outcome to assess the robustness of the subject under test [11].

We refer to the sentences produced by a user as *utterances*. They have an *intent*, which is the purpose of the query made by the user, and *entities*, which are the parameters of this query. For example, for the utterance "Add a 1 hour meeting for tomorrow at 2pm in room A1" the intent would be adding an event to the calendar, and the entities would be the time, date, duration and location of the event.

A schematic representation of the framework (see [19] for more details) is given in Figure 1, which works as follows. Original utterances are first processed by the agent in order to retrieve the *original intent*. These original utterances are then used to generate divergent utterances, which are in turn processed by the agent under test in order to retrieve the *divergent intents*. Based on those divergent intents, we can compute the *robustness* of an agent to a given intent. We define the *robustness* R to an intent i as $R_i = \frac{|C_i|}{|T_i|}$, where C_i corresponds to the divergent utterances correctly classified as being of intent i, and T_i designates all the divergent utterances of intent i, with the divergent utterances in both sets being generated from correctly classified original utterances only.

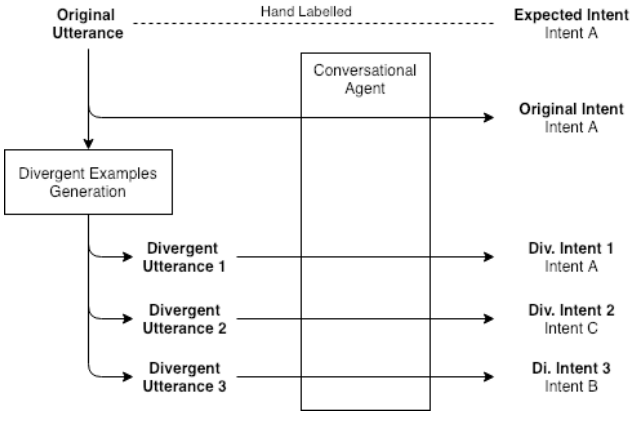

Figure 1. Schematic representation of the test framework in use. Original utterances are processed to generate divergent examples which help assess the robustness of the agent under test.

4. Divergent Example Generation

Paraphrasing in the context of conversational agent testing holds two additional constraints compared to other paraphrasing problems. Firstly, we focus on accuracy rather than coverage. We wish to minimise the number of false negative test cases, *i.e.* cases where the agent fails to correctly handle an utterance that a human agent would not understand either. Secondly, paraphrases should model various types of speech, for example by making use of informal vocabulary. The goal is to generate data that is as close as possible to utterances produced by human users, but also as diverse as possible, in order to anticipate real cases of misunderstanding.

These additional constraints lead us to favour a rule-based approach, as opposed to data-driven paradigms. We expect this type of approach to give us more flexibility and control with regards to the style of the output being produced, while not having to deal with data-sparseness issues that would otherwise be likely encountered. Moreover, it also seems important to have a fine-grain understanding of how paraphrases were generated in order to trace back misunderstandings.

We propose to generate divergent utterances by processing input data through **Lexical Substitutions**, which aims to change the vocabulary used in the utterance by either using generic and neutral synonyms (*Generic Lexical Substitutions*), as opposed to synonyms specific to a register of speech or a geographical region (*Targeted Lexical Substitutions*).

Note that we could use first a **Structural Modifications** step, similar to the work of Napoles et al. [15] but we believe that lexical paraphrases and other alterations to the surface forms of words may be more insightful when testing conversational agents. Indeed, agents are not capable of linking different words or written forms to the same concepts unless encountered in the training data or specifically handled, for example with a spelling corrector. While interesting, word ordering or the use of different non-key words might not have as much impact on the understanding capabilities. We therefore choose to focus on lexical substitutions as our main paraphrasing technique.

We assume that all input utterances are grammatically and syntactically correct with no spelling mistakes. We furthermore transform the input utterances to lower-case as a preprocessing step to avoid case sensitivity issues.

4.1. Lexical Substitutions

In the following we propose an approach to paraphrasing through lexical substitutions, that is we replace words occurring in a given input utterance with their synonyms. The goal of this paraphrasing is to introduce variations in the vocabulary, either by remaining generic or by targeting a specific type of English. We chose this approach because of its relative simplicity and little overhead compared to more complex solutions such as adapting a Machine Translation Engine.

4.1.1. Generic Lexical Substitutions. We follow the lead of Hassan et al. [10] and build our approach on these two successive steps:

1) Retrieval of possible synonyms for each word appearing in the input utterance, and generation of the candidate paraphrases by substituting the original word with its inflected synonyms.
2) Scoring of the candidate paraphrases yielded by the previous step in order to weed out bad candidates and provide a ranking of the remainders.

4.2. Generation of Candidate Paraphrases

The processing of each input utterance starts with tokenization and Parts-of-Speech tagging, using the Microsoft Cognitive Services API[2]. Dependency parsing using the Stanford Core NLP tools [12] is also performed in order to identify and retrieve dependencies of interest.

For each token obtained in the previous step, we skip over punctuation signs and stop-words. The remaining tokens are then lemmatized using the *Pattern* [21] library. In the case of phrasal verbs, the corresponding particle is retrieved based on the information obtained from the dependency parsing, and is included in the lemma form[3].

These lemma forms are then used to retrieve the possible synonyms from lexical databases, namely WordNet [14]. These synonyms are in turn inflected and inserted into the input utterance, yielding candidate paraphrases. The supported inflections include pluralization, singularization and conjugation. Indefinite article epenthesis[4] is also supported. These operations are handled by the *Pattern* [21] library.

Finally, we only consider paraphrases that are one editing distance away from the input utterance. We do not consider the simultaneous substitution of multiple lemma forms in the input utterance. This is done in order to to limit the number of generated candidate paraphrases and to maintain a better quality.

4.3. Scoring of Candidate Paraphrases

Many words have different meanings depending on the context in which they occur, and as such performing substitutions with synonyms will lead to nonsensical results. Filtering out poor paraphrases is therefore necessary, and we perform this task by adapting and combining some of the strategies presented by Hassan et al. [10]. The following strategies have been implemented:

- **Language Model (LM)** The fluency of candidate paraphrases is scored with a 5-gram language model. The probability of each candidate paraphrase is computed by the Language Model API provided by Microsoft's Cognitive Services platform.
- **Translation Pivoting (TP)** The input utterance is translated into a foreign language and then back into English, producing a pivoted sentence. Candidates whose inflected synonym appears anywhere in one of the pivoted sentences are given a score of 1, 0 otherwise. In our implementation, we rely on the Microsoft Translator Text API using French as pivoting language and requesting multiple translations to French and back to English in order to increase the diversity of the pivoted sentences.
- **Word Vectors (WV)** This strategy yields a higher score for candidate paraphrases whose inflected synonym has a strong lexical similarity with the input utterance. In order to assess the lexical similarity we rely on Word2Vec [13] using the Google News pretrained data[5] and Gensim [18]. The lexical score of a synonym is computed as the average of the cosine similarities between the inflected synonym and each word of the input utterance which is not a stop-word.
- **Web Search (WS)** Each candidate paraphrase is queried in a web search engine and scored based on how many hits were obtained. The search is performed using the Bing Web Search API.
- **Word-Sense Disambiguation (WSD)** For each word replaced in the input utterance, we retrieve its most probable sense using the Simplified Lesk algorithm provided by Pywsd [22]. Candidates whose synonym corresponds to the most probable sense of the replaced word are given a score of 1, 0 otherwise.
- **Lexical Frequency (LF)** This strategy scores candidates by how frequently its synonym has been encountered as a synonym for the different senses of the original word and in different lexical resources. The yielded score corresponds to how many times the synonym has been encountered.

Our previously described scoring methods are aimed at different aspects of what makes a good paraphrase. We measure the semantic relatedness of a synonym with the input utterance by using the Word-Sense Disambiguation and Word Vectors strategies, but also take into account the well-formedness of candidates with the Web Search and Language Model methods. Finally, Translation Pivoting is a high-precision but low-coverage strategy that allows us to identify the most plausible candidates, while Lexical Frequency encourages the use of synonyms with a broader sense. Since our scoring strategies are aimed at orthogonal aspects of the problem, it is expected that combining them together should provide overall better results.

The aforementioned strategies are combined together by first rescaling each of the metrics in the range $[0; 1]$, rescaled metrics which are then combined linearly as shown in equations 1 and 2. In these equations, s_m and s'_m represent the original and rescaled scores yielded by strategy m, λ_m is the weight of strategy m, c is the candidate paraphrase to score, C is the set of all paraphrases to score, and $s(c)$ represents the global score of candidate paraphrase c.

2. https://azure.microsoft.com/en-us/services/cognitive-services/

3. For example, the verb in the sentence "He made it up" lemmatizes to "make_up" rather than "make".

4. *I.e.* the indefinite article *a* becoming *an* before a vowel.

5. https://code.google.com/archive/p/word2vec/

TABLE 1. BEST WEIGHTS AND PRUNING THRESHOLDS FOR THE LEXICAL SUBSTITUTIONS SCORING TASK, OBTAINED BY FAVORING PRECISION OVER RECALL.

LM	TP	WV	WS	WSD	LF	Pruning
59	43	77	13	53	18	0.59

$$s'_m(c) = \frac{s_m(c) - \min_{x \in C}(s_m(x))}{\max_{x \in C}(s_m(x)) - \min_{x \in C}(s_m(x))} \quad (1)$$

$$s(c) = \sum_{i=1}^{m} \lambda_m * s'_m(c) \quad (2)$$

Candidates that obtain a global score lower than a predetermined threshold t are considered bad and are pruned from the final set of paraphrases.

The best weights λ_m and the best pruning threshold t are determined by a genetic algorithm which tries to maximise the amount of acceptable paraphrases being returned by our lexical substitutions. More specifically, the fitness function that the algorithm tries to maximise is the average *F-score* over all the input training utterances. The training data was collected by submitting all candidate paraphrases of randomly selected input utterances to a panel of fluent English speakers, with each judge being asked to select the candidates they deemed as acceptable paraphrases. The genetic search is moreover performed multiple independent times on random subsets of this training data in a process similar to *bagging*. The final weights and pruning thresholds are obtained by averaging the fittest individuals of each run in an attempt to limit over-fitting. These operations were performed using the *Deap* library [7]. The final weights are shown in Table 1, and were obtained by only taking precision into account, the recall rate having no impact on the fitness function. The precision yielded by these weights is 0.611, with a recall of 0.150 over all the training data.

4.3.1. Targeted Lexical Substitutions. We extend our approach presented in the previous section by relying on data provided by the Oxford Online Thesaurus. This thesaurus contains annotations for the register of speech[6] and the region of use[7] of proposed synonyms.

The process is thus the same as presented previously, with the exception that we use the Oxford Thesaurus rather than WordNet to retrieve synonyms and filter them according to the targeted register and/or region. We furthermore do not perform any word-sense disambiguation as we were unable to retrieve the definitions associated to the different synonyms. The weights and pruning threshold are the same as for the generic lexical substitutions.

Some examples of the divergent utterances generated by our strategies are showcased in Table 2.

6. For example, “informal”, “dated”, “archaic”...
7. For example, “American”, “British”, “West Indian”, ...

TABLE 2. EXAMPLE OF DIVERGENT UTTERANCES GENERATED BY THE DIFFERENT AFOREMENTIONED STRATEGIES. THE DIFFERENCES BETWEEN *original* AND *generated* UTTERANCES ARE UNDERLINED.

Lexical Substitutions (Generic)
Original: i need to book a flight. **Generated:** i need to reserve a flight.
Original: is my flight leaving as scheduled? **Generated:** is my flight departing as scheduled?
Lexical Substitutions (Informal)
Original: i want to cancel my ticket. **Generated:** i want to scrap my ticket.
Original: is it possible to cancel my booking. **Generated:** is it doable to cancel my booking.

5. Results

In this section we evaluate the ability of our conversational agent testing framework to identify understanding issues using paraphrases; we also study whether adding paraphrases to the language model used by the conversational agent improves its robustness.

5.1. Evaluating Lexical Substitutions based Paraphrasing

The goal of this section is to evaluate the performances of our lexical substitutions engine. In other words, we want to check how often a human judge would accept a lexical paraphrase that our system considers as valid. In the following we only focus on generic and informal lexical substitutions, that is targeted lexical substitutions with the informal register of speech as target. We thus carry out this evaluation by submitting randomly selected paraphrases to a panel of human judges who were asked whether or not a paraphrase should be accepted as valid. Judges were first presented with the candidate paraphrase before being shown the original sentence and asked to give their opinion.

This survey was conducted on a sample of 4 native or near-native English speakers. Out of 20 input sentences, 45 generic and 15 informal lexical paraphrases were generated and marked as valid by our system. The human acceptance rate of those paraphrases, according to our survey, is shown in Table 3. As illustrated by the overall lack of consensus, defining what constitutes a valid paraphrase is perhaps not as straight-forward as expected, some judges being seemingly lenient or on the contrary quite harsh. However, even if we are very lenient and consider paraphrases that have been accepted by at least one judge as valid, the human acceptance rate is at best approximately 50%. This implies that half of the paraphrases generated and marked as valid by our system are in fact not.

While those underwhelming results can be explained by the complexity of the task, they appear to be insufficient in the context of building a test framework. Rather than having a fully automated process, we therefore propose to shift our system towards *computer-assisted lexical paraphrase discovery*. That is, instead of having a pruning threshold above which candidate paraphrases are automatically considered

TABLE 3. HUMAN ACCEPTANCE RATE OF CANDIDATE PARAPHRASES CLASSIFIED AS VALID BY OUR SCORING SYSTEM, FOR GENERIC AND INFORMAL SUBSTITUTIONS.

	Generic	Informal
Unanimous	7/45 = 16%	3/15 = 20%
Majority	12/45 = 27%	6/15 = 40%
At least One	25/45 = 56%	8/15 = 53%

valid, we output the N best candidates according to our scoring strategies and ask human judges to manually filter those best candidates to only keep actually valid paraphrases. This new approach is very likely to help increase both precision and recall when compared to a fully automatic system, while still significantly reducing the human workload compared to fully manual task. With on average approximately 50 paraphrases being produced per input utterance on the data used for this survey, having a ranked and pre-filtered system also helps reduce the human workload. In the following of this document, we use $N = 10$. However, this computer-assisted system would still suffer from scalability issues, not mentioning the previously discussed consensus issue.

5.2. Case Study

In this section we analyse and discuss the use of our proposed testing framework [19] through a case study. We focus on evaluating the agent's ability to correctly identify the intent of an utterance and do not consider entity recognition. We conducted four tests using four utterance datasets, described below.

The agent under test for these experiments, which we refer to as *AirlineBot*, has been trained with the Microsoft LUIS platform[8] to handle typical customer queries that may be addressed to an airline. The different intents recognized by this agent are *BookFlight*, *CancelFlight*, *ChangeFlight*, *CheckTime*, *CheckWeather* and *FindFlight*.

To train and test this agent, we conducted an online public survey in which users were asked to type queries as if they were addressing a chatbot. We collected 357 utterances, each corresponding to one of the 6 intents. These utterances were manually filtered to only keep syntactically correct instances. This is an important step in the experiment to evaluate the proposed testing method while controlling the impact of paraphrase generation quality. We refer to this data as the *survey data*.

From the *survey data*, 10 utterances per intent were randomly sampled to serve as our *training dataset*. We limited the number of utterances to 10 to avoid overfitting the bot and ensure the same set size for each intent.

We created two paraphrase datasets comprised of paraphrases generated on the training data. More precisely, the paraphrases were yielded by generic lexical substitutions *(GS)* and informal lexical substitutions[9] (*IS*). We pass both types of paraphrases to the agent and analyse the output in terms of intent identification.

8. https://www.luis.ai/

9. That is targeted lexical substitutions with the informal register of speech as target.

TABLE 4. INTENT PRECISION OF *AirlineBot* AS MEASURED BY DIFFERENT SETS OF TESTS.

Intent	GS	IS	Test	All
BookFlight	**0.68**	**0.48**	0.82	0.83
CancelFlight	1	1	0.89	0.95
ChangeFlight	0.98	**0.73**	1	0.98
CheckTime	0.92	**0.43**	0.83	0.84
CheckWeather	1	1	1	0.97
FindFlight	1	1	1	0.95

TABLE 5. INTENT RECALL OF *AirlineBot* AS MEASURED BY DIFFERENT SETS OF TESTS.

Intent	GS	IS	Test	All
BookFlight	1	1	0.90	0.96
CancelFlight	**0.55**	**0.26**	0.80	0.89
ChangeFlight	**0.80**	1	0.90	0.88
CheckTime	0.88	1	1	0.95
CheckWeather	1	1	0.80	0.89
FindFlight	1	1	0.70	0.85

As these paraphrases were generated based on the training data, we also wanted to evaluate the performance of the agent once deployed (i.e. performance on unseen utterances). As such, we excluded the utterances in the *training dataset* from the *survey data* and used a set of 60 randomly sampled utterances from the filtered survey data (10 per intent) that we refer to as *Test*.In an attempt to approximate the performances of the agent once deployed, we also use the entire survey data as an additional (unbalanced) test set, referred to as *All*. We pass these utterances to the agent and analyse the output in terms of intent identification.

We present the results of each test in terms of precision and recall in tables 4 and 5. As we can observe, lexical substitutions seem able to uncover understanding weaknesses that would otherwise perhaps be undetected by a set of new test utterances. As indicated by the generic lexical substitutions set, many utterances requesting a flight cancellation appear to be understood as a booking request, leading to both a low recall for *CancelFlight* and *ChangeFlight*, and a low precision for *BookFlight*. Upon further inspection, the cause of the problem seems to be the use of words like "delete" and "invalidate" to refer to cancellation, and words like "switch", "shift" or "alter" to refer to modification requests. Whether or not those words should be accepted as valid synonyms in this context is up to debate, however the knowledge of these formulations causing problems can be useful to improve the performances of the agent, for example when handling non-native speakers.

Similarly to the generic substitutions, we can see that informal vocabulary seems to be poorly understood when it comes to cancellation requests, leading to the very low recall of *CancelFlight* and poor precision of *BookFlight*, *ChangeFlight* and *CheckTime*. Upon further inspection, it appears that the informal use of verbs like "nix", "scrub" or "axe" to refer to cancellation are the cause of the misunderstanding.

We now propose to analyze this conversational agent

TABLE 6. ROBUSTNESS PER INTENT OF *AirlineBot* TO DIFFERENT KINDS OF LANGUAGE VARIATIONS.

Intent	GS'	IS'
BookFlight	1	0.78
CancelFlight	**0.72**	**0.36**
ChangeFlight	**0.76**	**0.75**
CheckTime	0.97	1
CheckWeather	0.84	0.89
FindFlight	1	1

TABLE 7. TOTAL NUMBER OF TRAINING UTTERANCES PER INTENT AND ENRICHED TRAINING DATA.

Intent	GS+IS	Additional
BookFlight	50	25
CancelFlight	90	25
ChangeFlight	72	25
CheckTime	38	25
CheckWeather	73	25
FindFlight	37	25

TABLE 8. PRECISION OF *AirlineBot* ON THE *All Filtered* TEST DATA, SETS PER INTENT AND TRAINING DATA.

Intent	Original	Additional	GS+IS
BookFlight	0.68	0.68	0.63
CancelFlight	0.90	0.82	0.78
ChangeFlight	0.99	1	0.92
CheckTime	0.63	0.69	**0.57**
CheckWeather	0.88	1	1
FindFlight	0.89	0.89	0.82

TABLE 9. RECALL OF *AirlineBot* ON THE *All Filtered* TEST DATA, SETS PER INTENT AND TRAINING DATA.

Intent	Original	Additional	GS+IS
BookFlight	0.90	0.90	0.90
CancelFlight	0.86	0.86	0.86
ChangeFlight	0.87	0.88	0.85
CheckTime	0.83	0.92	**0.67**
CheckWeather	0.70	**0.90**	0.70
FindFlight	0.76	0.81	**0.67**

using the proposed robustness metric. Assuming all training utterances are correctly classified by the final agent, the recall on the *training paraphrases* and the robustness to the training utterances actually correspond to the same metric. We therefore perform this analysis by generating adversarial examples based on the previously sampled test data. In order to assess this robustness, we only keep the already correctly classified utterances that are in turn paraphrased using generic lexical substitutions (*GS'*) and informal lexical substitutions (*IS'*).The robustness results for each intent are presented in Table 6.

As previously highlighted by our training paraphrases, we can notice the same understanding difficulties for the *CancelFlight* and *ChangeFlight* intents, and for similar reasons.

In conclusion, testing a conversational agent with paraphrases can potentially help uncover weaknesses that would otherwise possibly go unnoticed with regular testing data. This observation appears moreover to hold true for paraphrases generated from both training and test data.

5.3. Enriching the Training Data

In this section we investigate the use of paraphrases to enrich agent training data. We compare three sets of training data: (*a*) the *original* dataset as used in the previous section, (*b*) the *original* dataset enriched with the paraphrase datasets described in the previous section which we refer to as *GS+IS*, and (*c*) the *original* dataset enriched with 15 additional utterances per intent sampled from the *survey* data to which we refer to as *additional*.The sizes of the new training sets are shown in Table 7.

We first analyze the impact of the new training datasets on the performance of the agent to detect any potential negative impact. In order to do so, we compare the precision and recall on all the utterances from the *survey data* that are not used as training data in any of the aforementioned sets. We refer to this test set as *All Filtered*. As shown in tables 8 and 9, when compared to its original performances and trained on *GS+IS*, our agent seems to be only slightly negatively impacted, with the exception of the *CheckTime* intent which sees a bigger dip in both precision and recall. When compared to the performances yielded by an agent trained on the *additional* data, we can however see a bigger difference in terms of recall for the *CheckWeather* and *FindFlight* intents. Those results seem to indicate that our enriched training data leads neither to an improvement nor a significant decrease in performances for this test dataset.

We compare the robustness of the agent for the different sets of training data (see Table 10). We can observe that some issues we previously identified can indeed be corrected when using enriched training data, leading to an improved robustness on some intents, such as *CancelFlight* for informal substitutions or *ChangeFlight* for generic substitutions. It appears, however, that while simply using more training data to solve those specific understanding issues can lead to a slight robustness improvement, such as for *CancelFlight*, using enriched training data seems to be the most efficient way to address cases that have been identified as problematic.

While we are not able to witness any improvements when testing our agent on the generic utterances present in *All Filtered*, we do not witness any major decrease in performances either. This could suggest that our agent is more robust to the phenomenons not explicitly encountered in the available data, while mostly retaining the overall same performances for the more commonly encountered cases. In a nutshell, training paraphrases should perhaps not be considered as an alternative to more training data, but rather as a complimentary tool that helps achieve better coverage when it is sparse.

6. Conclusion

This work demonstrates the multiple ways paraphrases can be of use when trying to build conversational agents that

TABLE 10. ROBUSTNESS OF *AirlineBot* PER INTENT, TRAINING DATA AND LANGUAGE VARIATION.

Training Data	GS'	IS'
BookFlight		
Original	1	0.78
Additional	0.91	0.78
Enriched with GS+IS	0.79	0.89
CancelFlight		
Original	0.72	0.36
Additional	0.87	0.52
Enriched with GS+IS	0.91	**0.95**
ChangeFlight		
Original	0.76	0.75
Additional	**0.57**	0.75
Enriched with GS+IS	**0.96**	0.75
CheckTime		
Original	0.97	1
Additional	1	1
Enriched with GS+IS	1	1
CheckWeather		
Original	0.84	0.89
Additional	0.88	0.89
Enriched with GS+IS	1	1
FindFlight		
Original	1	1
Additional	1	0.86
Enriched with GS+IS	1	1

are robust to language variation. It appears that paraphrases can be a valuable source of knowledge when trying to build robust conversational agents, especially when data is sparse.

As we illustrated with our testing framework, we are able to uncover an agent's understanding weaknesses thanks to paraphrases generated from both its testing and training data. While paraphrases generated from training utterances can be useful to uncover understanding issues caused by the vocabulary choice, it seems nonetheless more appropriate to generate paraphrases from independent testing data, as we are not only able to detect the same shortcomings as before but also new ones caused by writing errors. Moreover, we also showed that enriching the training data of an agent with paraphrases could help cope with the previously identified issues, with seemingly no major negative impact in terms of precision and recall on the same test data. This suggests that paraphrases should perhaps not be considered as an alternative to more data, but as a complimentary tool that can help increase coverage of some phenomenons.

Acknowledgement

This work was supported with the financial support of the Science Foundation Ireland grant 13/RC/2094.

References

[1] Regina Barzilay and Kathleen R McKeown. Extracting paraphrases from a parallel corpus. In *ACL*, pages 50–57, 2001.

[2] David L Chen and William B Dolan. Collecting highly parallel data for paraphrase evaluation. In *ACL-HLT*, pages 190–200, 2011.

[3] Tsong Y Chen, Shing C Cheung, and Shiu Ming Yiu. Metamorphic testing: a new approach for generating next test cases. Technical report, Technical Report HKUST-CS98-01, Department of Computer Science, Hong Kong , 1998.

[4] Tsong Yueh Chen, Fei-Ching Kuo, Huai Liu, Pak-Lok Poon, Dave Towey, T. H. Tse, and Zhi Quan Zhou. Metamorphic testing: A review of challenges and opportunities. *ACM Comput. Surv.*, 51(1):4:1–4:27, January 2018.

[5] Timothy Chklovski. Collecting paraphrase corpora from volunteer contributors. In *K-Cap*, pages 115–120, 2005.

[6] Ilker Koksal for BotAnalytics. These are the most important chatbot metrics to track, 2017. Accessed: 2017-10-13.

[7] Félix-Antoine Fortin, François-Michel De Rainville, Marc-André Gardner, Marc Parizeau, and Christian Gagné. DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, 13:2171–2175, 2012.

[8] Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. Ppdb: The paraphrase database. In *ACL-HLTL*, pages 758–764, 2013.

[9] Ankush Gupta, Arvind Agarwal, Prawaan Singh, and Piyush Rai. A deep generative framework for paraphrase generation. *arXiv preprint arXiv:1709.05074*, 2017.

[10] Samer Hassan, Andras Csomai, Carmen Banea, Ravi Sinha, and Rada Mihalcea. Unt: Subfinder: Combining knowledge sources for automatic lexical substitution. In *SemEval*, pages 410–413, 2007.

[11] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.

[12] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *ACL*, pages 55–60, 2014.

[13] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *ACL-HLT*, volume 13, pages 746–751, 2013.

[14] George A Miller. Wordnet: a lexical database for english. *CACM*, 38(11):39–41, 1995.

[15] Courtney Napoles, Chris Callison-Burch, and Matt Post. Sentential paraphrasing as black-box machine translation. In *NAACL*, pages 62–66, 2016.

[16] Ellie Pavlick, Pushpendre Rastogi, Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. Ppdb 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. 2015.

[17] Chris Quirk, Chris Brockett, and William Dolan. Monolingual machine translation for paraphrase generation. In *EMNLP*, 2004.

[18] Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. http://is.muni.cz/publication/884893/en.

[19] Elayne Ruane, Théo Faure, Ross Smith, Dan Bean, Julie Carson-Berndsen, and Anthony Ventresque. Botest: a framework to test the quality of conversational agents using divergent input examples. In *IUI*, pages 64:1–64:2, 2018.

[20] Yusuke Shinyama, Satoshi Sekine, and Kiyoshi Sudo. Automatic paraphrase acquisition from news articles. In *ACL-HLT*, pages 313–318, 2002.

[21] Tom De Smedt and Walter Daelemans. Pattern for python. *Journal of Machine Learning Research*, 13:2063–2067, 2012.

[22] Liling Tan. Pywsd: Python implementations of word sense disambiguation (wsd) technologies [software]. https://github.com/alvations/pywsd, 2014.

[23] Wei Xu. *Data-driven approaches for paraphrasing across language variations*. PhD thesis, 2014.

[24] Shiqi Zhao, Cheng Niu, Ming Zhou, Ting Liu, and Sheng Li. Combining multiple resources to improve smt-based paraphrasing model. In *ACL*, pages 1021–1029, 2008.