A Robust Spoken Q&A System with Scarce In-Domain Resources

Luis Fernando D'Haro, Seokhwan Kim, Rafael E. Banchs
Institute for Infocomm Research - A*STAR
1 Fusionopolis Way, Connexis South
Singapore, 138632
E-mail: luisdhe@i2r.a-star.edu.sg Tel: +65-6408 2146
E-mail: kims@ i2r.a-star.edu.sg Tel: +65-6408 2766
E-mail: rembanchs@ i2r.a-star.edu.sg Tel: +65-6408 2802

***Abstract*— Nowadays there is an increasing interest on deploying spoken conversational agents to provide ubiquitous Question and Answering information to customers about corporate services and commercial products and supporting different users' devices such as PC desktops or mobile phones. Unfortunately, creating an accurate system requires a lot of handwork, where developers must consider several factors such as the performance of the ASR system, the presence of typos in the transcribed queries, the large number of possible variations to ask for the same information using different sentences, or the subtle differences that could exist between similar, but semantically different, questions. In this paper, we propose a methodology for quickly creating robust spoken-based conversational agents with very low resources.**

Our solution only requires few hand-made query samples, which are automatically expanded to deal with the use of different synonyms and wordings; next, spoken queries are automatically generated using a TTS system and then the audio files are corrupted to simulate different noise conditions and environments that the final users can experiment when they query the system using their voice with means of their mobile devices or by using a kiosk. Then, these audio files are subsequently transcribed using a general purpose ASR which produces an n-best list of recognized results that is first used to retrieve relevant documents and then re-ranked in order to select the final answer.

Our tests on a set of 21 different topics proves that our proposal can get a 13% absolute better accuracy than a standard IR using an index with only in-domain answers and 6.3% better than a system including millions of negative out-of-domain candidate answers which is what it is expected for a scalable system.

I. INTRODUCTION

In recent years, the number of companies including conversational agents on their websites has increased continuously. These agents are responsible for providing 24/7 service to customers looking for information available on the website, or for answering common questions about the services provided by the company (e.g. prices, promotions, deliveries, products availability, troubleshooting, FAQ, etc.). Unfortunately, the deployment of such systems is not an easy task as it requires a continuous development process where the system is improved as much as the users use it and the company decides to increase the amount of topics and answers it can handle. In addition, the system must provide accurate answers on a particular domain for which scarce or none previous examples are available, and in spite of the different words users use, possible errors in the query, or in spite of the similarities between the different topics and answers that the system is able to answer.

This kind of "cold start" problems have been studied in the literature, especially in the context of recommendation systems for movies [1], books [2], or social networking [3], etc. In these cases, three different approaches are taken: collaborative filtering, content-based filtering, or a combination of both [4]. In the first case, the recommendations are based on collecting and analyzing a large amount of information about similar existing users/items. The advantage is that the system does not require a complete understanding of the user/item just finding similar users/items; in the second case, the recommendation is based on specific user's profile information or item description. Here, the recommendations are based on similar items that the user liked in the past or that is viewing in the moment.

Unfortunately, in the case of spoken Q&A systems, the cold start problem poses more difficulties since several modules must be fine-tuned in order to improve the overall performance of the system, e.g.: a) the ASR engine (including pronunciation dictionaries, as well as acoustic and language models) used to transcribe the user utterances, b) the content of the index or database used to retrieve the relevant information, and c) the dictionary of relevant keywords the system must be able to understand. In this paper, we focus on proposing robust solutions allowing the system to understand spoken queries automatically transcribed to retrieve relevant answers (i.e. high precision), and to deal with speech recognition errors, misspellings and out-of-vocabulary words.

This paper is organized as follows: in section II we describe the data collected for creating the Q&A system for an important corporate site in Singapore; in section III, we describe the proposed domain-independent architecture; in section IV, we present our results and finally in section V we show the conclusions and future research.

II. DATA DESCRIPTION

In order to develop our system, we were given with a set of 21 different canonical questions and their corresponding answers. Since the goal of the proposed system is to provide a very high Precision@1 and considering that there were not any available logs to analyze how users could ask for information to the system, we manually generated 533 question paraphrases from the original set of canonical questions (see Table 1) and then we classify them as positive (330) or negative (203), using as criteria that the paraphrase would be considered as positive if its answer is given by the same answer provided for the canonical question, and as negative in case it was not. Therefore, some of these paraphrases were actually valid paraphrases of the canonical questions, while others were invalid distractors. Consider for instance the following canonical question "Can I bring my pet to your corporate offices?" with the corresponding answer: "Yes, however the pet must be caged up or kept in a leash". For this case, some examples of valid positive question paraphrases are:

- "Is there any restriction to bring my pet along?"
- "Is it possible to take my dog to your place?"
- "Are cats allowed in your installations?"
- "Tell me if there are restrictions to go with my cute puppy to your building"

While negative examples are:

- "Can I buy a pet fish in your place?"
- "Is there any pet veterinarian in the building?"
- "Which pets are forbidden to bring?"

Notice from these paraphrases that, although some of the positive examples could require a minor rephrasing of the answer for it to be coherent with the question, the answer can still be considered as appropriated and useful to the user. On the other hand, it is clear that the classification task is very difficult since the negative examples belongs to the same topic of questions (i.e. pets), and could be formulated using very similar words to the ones occurring in the positive examples. For the case of negative examples, the expected behavior of the system is to inform the user that there is not an available answer for that particular question and offer a sorted list of related questions.

Taking into consideration the variability of the paraphrases, an important desired characteristic of the proposed system is that it must be able to allow the use of synonyms and hypernyms (which we explain in section C). On the other hand, since a high accuracy in the answers is required, we included a question-type classifier based on regular expressions that automatically identifies the type of factoid query posed by the user (i.e. comparison, how many, how much, how to, need request, why, what, when, where, which, who, and none) reducing the possibility of the system to provide an answer that cannot match the type of query.

Finally, since our proposed system must be scalable with regards to new questions and topics, as well as dealing with many other kind of negative examples, we extended our set of negative examples for the original question dataset (i.e. the 21 canonical questions) with Q&A pairs from the Webscope L6 Yahoo! Answers Comprehensive Questions and Answers version 1.0. This corpus is a collection of questions and answers posted by people at the Yahoo! Answers website as of 10/25/2007. It includes a total of 4,483,032 questions and their corresponding answers plus a small amount of metadata, i.e., which answer was selected as the best answer, and the category and sub-category that was assigned to each question. Since some of the questions are not in English, we filtered out non-English questions and retained 3,895,406 queries.

Table 1. *Statistical information about the number of positive and negative paraphrases for each of the 21 canonical questions.*

Class	C	#	C	#	C	#	C	#	C	#
Pos	1	19	6	17	11	15	16	15	21	14
	2	18	7	17	12	17	17	15		
	3	19	8	13	13	13	18	12		
	4	16	9	20	14	12	19	15		
	5	16	10	10	15	20	20	16		
Neg	1	13	6	8	11	12	16	11	21	8
	2	11	7	10	12	10	17	9		
	3	11	8	10	13	10	18	9		
	4	9	9	9	14	9	19	10		
	5	8	10	13	15	7	20	7		

III. GENERAL PURPOSE ARCHITECTURE

Our current system takes advantage of previously deployed modules and algorithms. For instance, from [5] we borrowed the idea of performing a re-ranking over a set of pre-selected candidates. Therefore, in the current system, the first step (see section 3.4) is to retrieve a list of candidate answers from the index created with the canonical answers, the positive and negative paraphrases and also out-of-domain answers (i.e. Yahoo! Answers); then, the list is re-ranked using a SVM classifier (see section 3.5) that chooses the final system answer. From [6], we use the module that deals with transcription misspellings as well as the runtime platform that connects the client (a web browser or a mobile app) and the web server using web sockets. Below more details of the new modules are given.

A. Automatic generation & recognition of spoken queries

One important factor we took into account for deploying this spoken-based Q&A system is that it will be extensively used on public places, where different kinds of noise and environmental conditions are likely to occur. Since collecting audio data for all the possible canonical and paraphrase questions as well as environment conditions is a time and expensive process, we decided to simulate all user's spoken

queries by converting the hand-made and automatic paraphrases text into voice by using Google TTS. Then, in order to simulate the different environment and noise conditions, the TTS audios were corrupted by adding different kind of noises at a several signal-to-noise level ratios (SNR). Although this procedure does not guarantee that the Q&A system is tested against the same distribution of ASR errors produced when using real speech, the advantage is that it allows us for quickly testing the deployed system under different conditions, while true on-site audio recordings are collected. For future deployments, we plan to validate this methodology by contrasting our current simulated results with those obtained using real speech.

As mentioned before, in order to test the robustness of the ASR in the presence of different noise conditions and levels, we experimented adding noise to the generated speech data using 5 different signal-to-noise ratios: clean, 15 dB, 10 dB, 5 dB, and 0 dB. In addition, we also considered two different kinds of noises: white and environmental. For the last one, we combined our TTS generated files with a set of creative commons audio files containing noise recordings of different public places (shopping centers, restaurants, airport, halls, etc.). The mixed files were generated using FaNT toolkit [8].

Table 2. *Google ASR WER results on different audio conditions. Only results for successfully transcribed queries are reported. (w.Av. means weighted average).*

SNR (dB)	White Noise				Environment Noise			
	No Q	Min.	Max.	1-Best	No. Q	Min	Max	1-Best
0	4	96.7	96.7	96.7	109	60.4	77.4	67.4
5	397	46.1	68.7	55.1	481	30.0	52.1	36.5
10	531	18.9	42.3	25.6	531	10.5	32.9	15.1
15	532	9.8	32.9	14.7	533	5.5	27.7	10.4
Total w.Av.	1464	23.2	46.2	29.8	1654	17.8	39.7	23.3

Clean				
	No. Queries	Min. WER	Max. WER	1-Best
Total	533	5.8	28.8	10.1

B. ASR N-Best lists

After generating the clean and noisy speech files, we transcribed the audio by means of a general purpose ASR engine. Considering the study done in [9], for our experiments we also decided to use Google Speech ASR which provided us with a list of 5-10 best candidates and a confidence score for the first candidate. The purpose for not using a domain-specific ASR was to account for the possibility of final users using their mobile built-in speech recognition system.

After obtaining the ASR transcriptions we calculated the WER for the N-Best list considering three cases (see Table 2): using only the 1-best result, considering the best transcription in the list (min WER), and considering the worst transcription (max WER). From these results we can see that in most of the cases the n-best lists provided good transcriptions even with low levels of SNR. However, we can also see that not always the 1-best option provides the lowest WER. This fact motivated us, in the experiments reported in section B, i.e. to query the index with a combination of all the transcriptions in the corresponding n-best list. On the other hand, since it could happen that the ASR recognition result is not good (e.g. the ASR confidence is low or the number of candidates is low) the system should be able to prompt the user for asking the question again rather than starting the retrieval process. In order to allow the system to be aware of the quality of the input audio and the n-best list, and considering that the Google ASR only provides a confidence score for the first candidate, we decided to extract a set of metrics that can be tested against a predefined threshold to ask the user to repeat the question or to start the retrieving process. Following the features proposed in [11], and including some new features, we extracted the following ones:

- The average perplexity of the n-best list candidates. Here the LM was trained on the one-billion-lm-corpus [12] using KenLM toolkit [13].
- First and Second Hypothesis LM N-gram perplexity.
- The drop in the LM perplexity between the first and second hypothesis in the n-best list.
- Average N-best Purity of all words in the N-best list
- The percentage of words across all N-best list hypotheses which have an N-best purity of greater than one half.Min and Max number of words in the N-best Hypothesis.
- Total Number of Words and vocabulary in the n-best list.
- Number of sentence hypotheses in the N-best list. This number was usually 5 but in 7.5% of the times it was less (especially on noisy files).
- Confidence score: Value returned by the ASR.

Although in our current implementation we could not combine all these features to generate the final confidence score, we are considering applying re-ranking techniques as the one proposed in [10].

C. Automatic creation of alternative sentences

One of the problems that the runtime system must face is the huge variability of valid paraphrases for each canonical question (i.e. generated by using synonyms, hyponyms or hypernyms). In the literature, we can find several algorithms for creating paraphrases combining different levels of abstractions and techniques [14][15]. Our proposed algorithm first extracts the Part-Of-Speech (POS) information and lemmatization for each word in the canonical question. Then, it looks for hyponyms and hypernyms in WordNet 3.0 [16][17] whose similarity (calculated as the shortest path that connects the senses in the is-a-hypernym/hyponym taxonomy) is above a predefined threshold (in our experiments it was set to 0.6). Then, new alternative words are added by extracting

semantically and contextually related terms from Word2Vec [18][19] and Glove [20]. Finally, more candidates are further obtained for each word by using a Thesaurus dictionary considering both the word itself and its POS tag. Once all the possible expansions has been generated, we filter the candidates by taking only those paraphrases whose perplexity is not higher than a relative ratio between it and the perplexity of the original sentence (in our experiments we set the ratio to be 20%). Here, we used the same language model as in section 3.2. As a result we obtained 439 sentences from the original 21 canonical questions. Below we show some paraphrases generated from the canonical questions: Where can I post my letter?

- Expanded: Where can I mail my letter? (POS)
- Expanded: Where can I post my correspondence? (POS)
- Expanded: How can I post my letter? (POS)
- Expanded: Where can I publish my letter? (NEG)

In the example, we can see that the expansion module is not perfect since it is possible to generate false positive (or false negative) examples that the designer must manually reassign to the corresponding category. However, what we have observed is that, in average, the number of true-positive and true-negative automatic paraphrases are higher than the false-positive and false-negative cases, therefore the manual reassign procedure could be omitted.

D. Index and search engine

Currently we can find several open-source tools for indexing and searching such as Solr/Lucene, Sphinx, Whoosh, or Lemur. Most of them make use of vector space models to retrieve relevant documents based on the similarity between the vectors generated from the words used in the given transcribed query and the words occurring in the indexed documents. The advantage of these engines is that they provide all the algorithms and tools required to process the queries (e.g. tokenization, lemmatization, spelling correction, fuzzy-text search, weighting of terms in the query, Boolean operators, etc.), as well as to index the documents and to conduct a fast search. In our implementation, we decided to use Lucene to take all the advantages of it as both a baseline system and as a sub-component of our proposed system.

Traditionally, a search engine for a particular domain is built only with in-domain documents. However, given the too few domain-relevant questions to be indexed in our case, this approach was not appropriated since it would not generated appropriated term-vectors and relevance weights. To solve this issue, our small set of in-domain canonical questions was stored in the index along with a large number of negative examples from Yahoo! Answers. For a given transcribed input query, the constructed index is used for retrieving a set of candidate questions. In our implementation, each test query is constructed by concatenating the corresponding N-best ASR hypotheses for each manually generated question paraphrases. The simplest way to get the system output is just to take the top-ranked candidate from the Lucene result, and the answer is decided based on the source category of it, as follows:

$$f(x) = \begin{cases} y & if\ d_{x,i} = c_y \\ NIL & otherwise \end{cases} \quad (1),$$

where x is an input query, $d_{x,i}$ is the i-th item in the ranked list from Lucene and c_y is the y-th canonical question in $C=\{c_1,...,c_{21}\}$. This approach is expected to produce relatively precise outputs by considering the queries which have more similar surface forms to the negative examples than to the in-domain questions as out-of-domain cases. However, the coverage of the method should be limited, because this process is performed only considering the surface form-level similarities. Thus, it will fail to retrieve the relevant answer when an input query has different surface forms from its corresponding in-domain example even if they have the same meaning to each other. This limitation could be worse with speech inputs, because ASR has the possibility of incorrect recognition for some important terms which play a crucial role in the retrieval.

E. Document Re-ranking algorithm

To enhance the recall of the proposed approach, with minimized precision loss, we propose a re-ranking stage. In this second stage, the Lucene results are re-ranked with a supervised learning to rank model trained on the ASR hypotheses for the manually created paraphrases described in Section II. In this work, we used SVM^{RANK} [21] which is a pairwise ranking algorithm learned from the ranked lists. For each pair of a query x in the training data and its i-th candidate in the Lucene results $d_{x,i}$, the ranking score $s_{x,i}$ is assigned as:

$$s_{x,i} = \begin{cases} 1 & if\ x \in P_y^+\ and\ d_{x,i} = c_y \\ 0 & otherwise \end{cases} \quad (1)$$

Where P_y^+ is a set of positive paraphrases for the y-th canonical question. The candidate list with the scores provides the relative orders for a given query, and it is converted into a set of pair wise constraints which are trained by SVM^{RANK} with the following features:

- N-gram Similarity: Cosine similarity between n-gram vectors from the surface forms of x and $d_{x,i}$, where n is 1, 2, and 3.
- Keyword Similarity: Cosine similarity between keyword vectors extracted from x and $d_{x,i}$.
- Question Type Similarity: Cosine similarity between question type vectors extracted from x and $d_{x,i}$.
- Phonetic Similarity: Weighted sum of phonetic similarities computed based on Levenshtein distances between metaphones of aligned word pairs in x and $d_{x,i}$. The aligned word pairs are selected by a maximum cardinality matching algorithm [22] on the bipartite graph between both keyword sets weighted by the similarities. The purpose of using this phonetic similarity was to

especially deal with the problem that the ASR was not able to correctly recognize some proper nouns or dialectal variations appearing in the questions, and unfortunately, we could not change the ASR vocabulary. However, in this case, we noticed that although the actual word was not recognized, a phonetically similar name was. For instance, consider the following canonical question "Where are the business centers located?", in this case for a speech audio with 10 dB noise level, one of the ASR hypothesis was: "Where is a business enters locator". Here, we see that the words centers and enters, and located and locator are phonetically similar therefore reinforcing the correct transcription form.

- Semantic Similarity: Weighted sum of semantic similarities computed based on word embeddings generated by Word2Vec. The same graph matching algorithm as the phonetic similarity is used but with semantic similarity for each aligned word pair.

The weight value for each term or n-gram unit in computing cosine similarities and weighted summations is the corresponding TF-IDF score computed over the whole data collection stored in the index.

IV. Experiments and Results

A. Experimental setup

To demonstrate the effectiveness of our proposed architecture, we performed experiments on the data described in Section II following a two-fold cross validation process. The first set consisted of the first 11 canonical questions while the second one comprised the remaining 10. In addition, the Yahoo! Answers database was also split into two sets containing 1,947,704 and 1,947,703 questions respectively. For each fold, we built the following three different types of systems:

1) Baseline: Index with only the 21 canonical questions.
2) Our proposed architecture without re-ranking and with/without sentence expansions. In the last case, the index contains not only canonical question and Yahoo! Answers data, but also automatically created sentences described in Section C.
3) Our proposed architecture using re-ranking trained with various combinations of features: a) n-grams, b) keywords, c) question type, d) phonetic, e) semantic.

Given a query, the top 100 candidate results from the full index were retrieved using Lucene with its default configuration and the standard analyzer. In addition, the candidates of a constrained search only on the set of canonical questions were also included along with a dummy candidate for the NIL cases. As training datasets for our proposed re-ranking approaches, the manually expanded and annotated ASR hypotheses (section II) were also divided into the same two folds as the other resources according to their original canonical questions before the expansions. Then, the ranking models for each fold were trained using SVM^{rank} toolkit with the features described in section E. For cross-validation, the actual re-ranking on the candidates for a given query belonging to one of two sets was done with the model trained on the other set. Thus, we finally have two models for each system: one is from the first set of canonical and negative questions and the other is from the second set of the datasets. Then, the model trained with the first set was used to retrieve the answers to the queries in the second set, and vice versa.

The system performance was evaluated by comparing to the manual annotations with micro-averaged precision, recall, and F-measure over the 21 canonical question categories. Additionally, average accuracy was also computed by considering the task as a 22-class multi-class classification problem. The 22 classes correspond to the 21 canonical questions plus the NIL category.

B. Experimental Results

Table 3 compares the performances among different approaches with various feature combinations. The baseline system only with in-domain collection achieved higher recall than the other systems, but its very low precision draws down the overall performance of the system in terms of accuracy. On the other hand, the use of a large amount of out-of-domain questions contributed to improve the precision and the classification accuracy of the second system compared to the baseline. However, it still failed to achieve higher performance in F-measure because of a falling-off in recall. This lack of coverage problem had been mitigated with re-ranking models. Especially, the model incorporating all the proposed features outperformed the baseline by 14% absolute in accuracy and 5% absolute in F-measure. This result is due to a large increase in recall (as compared to the system without re-ranking), while the drop in precision between two cases is not big.

Table 3. *Results on Precision, Recall, F-Measure, and Accuracy for the different systems*

	Without Automatic Expansions				With Automatic Expansions			
	P	R	F	A	P	R	F	A
1	43.01	**65.88**	52.05	42.77	-	-	-	-
2	**73.63**	26.33	38.79	49.43	**75.32**	28.51	41.37	50.82
3A	71.59	34.30	46.38	52.35	67.73	44.44	53.67	54.60
3B	72.22	36.43	48.43	53.40	66.84	45.02	53.80	54.13
3C	71.01	37.06	48.70	53.12	67.64	46.40	55.04	55.04
3D	69.84	38.83	49.91	53.15	68.41	50.18	57.89	**56.63**
3E	70.89	47.78	**57.08**	**56.82**	66.04	**51.91**	**58.13**	55.78

Adding the automatically expanded sentences into the index had a beneficial influence into obtaining further coverage with respect to the previous systems. For every feature combination, the expansions contributed to achieve significantly higher recalls. Some of the automatically generated expansions might be noisy, which caused some reduction in the observed precisions. However, the much

higher gains in recall produced better performances in F-measure, without any additional human effort. Finally, the model with all the features, as well as with the expanded sentences, produced the best result in F-measure, which is 6% absolute higher than the baseline.

V. Conclusions & Future Work

In this paper we have described a robust spoken question and answering system which allows users to access FAQ information about corporate services and facilities. The system has been tested against automatic generated spoken queries containing different levels and types of noise in order to simulate spoken queries done at crowded public places such as train/buses stations, airports, or shopping centers and recognized using a state-of-the-art general purpose ASR. The proposed architecture first takes into account the uncertainty on the recognized utterances by combining the information provided by ASR N-best lists. The system uses the ASR outputs to retrieve candidate responses from an index, which is based on a standard Lucene search engine. The retrieved candidates are then re-ranked by using a pair-wise algorithm, which has been found to achieve a 56.82% of accuracy (~6% absolute better than the traditional baseline system). The main advantage of the proposed system is that it is able to deal with very similar questions and providing only answers to those that are semantically similar to the canonical questions (high precision), but also allows for searching similar questions (high recall).

As future work, we first propose to improve the algorithm for the automatic creation of question sentences by using paraphrasing techniques such as the ones proposed by [22][23]. Finally, we also want to improve the proposed re-ranking algorithm by using semantic triplets, deep parsing, and question/answers type agreement as additional features

Acknowledgment

This project has been supported by the SERC industrial project (EC-2013-045). We also thank the Yahoo! Labs Webscope Team for providing us access to the L6 Yahoo! Answers Comprehensive Questions and Answers version 1.0 dataset.

References

[1] J. Bobadilla, F. Ortega, A. Hernando, and J. Bernal. “A collaborative filtering approach to mitigate the new user cold start problem,” *Journal Knowledge-Based Systems*, Vol. 26, Feb. 2012, pp. 225-238.

[2] A. Elbadrawy and G. Karypis. "Feature-based similarity models for top-n recommendation of new items," *Department of Computer Science, University of Minnesota, Minneapolis, Minnesota, Tech. Rep* (2013): 14-016.

[3] X. Zhou, Y. Xu, Y. Li, A. Josang, and C. Cox. “The state-of-the-art in personalized recommender systems for social networking,” *in Artificial Intelligence Review*, Vol. 37(2), Feb. 2012, pp. 119-132.

[4] A. Gunawardana and C. Meek. “A Unified Approach to Building Hybrid Recommender Systems,” *in Proceedings of the third ACM conference on Recommender systems*, pp. 117-124. ACM New York, NY, USA, 2009.

[5] R. Banchs, and H. Li. "IRIS: a chat-oriented dialogue system based on the vector space model," *in Proceedings of the ACL 2012 System Demonstrations. Association for Computational Linguistics*, 2012, pp. 37-42.

[6] L. F. D’Haro, S. Kim, K. H. Yeo, R. Jiang, A. I. Niculescu, R. E. Banchs, and H. Li. "CLARA: a multifunctional virtual agent for conference support and touristic information," *in Proceedings International Workshop on Spoken Dialog Systems, IWSDS15*, Busan, South Korea, Jan 11-13, 2015.

[7] J. Yamagishi, T. Nose, H. Zen, Z. Ling, T. Toda, K. Tokuda, S. King, S. Renals. “A Robust Speaker-Adaptive HMM-based Text-to-Speech Synthesis,” in *IEEE Audio, Speech, & Language Processing*, vol.17, no.6, pp.1208-1230, August 2009.

[8] G. Hirsch, “FaNT - Filtering and Noise Adding Tool” *Tech.Rep., Niederrhein University of Applied Sciences*. Available at http://dnt.kr.hs-niederrhein.de/

[9] F. Morbini, K. Audhkhasi, K. Sagae, R. Artstein, D. Can, P. Georgiou, S. Narayanan, A. Leuski, and D. Traum. "Which ASR should I choose for my dialogue system," *In Proceedings of the 14th annual SIGdial Meeting on Discourse and Dialogue*, pp. 394-403. 2013.

[10] R. Basili, E. Bastianelli, G. Castelluci, and D. Nardi. “Kernel-based discriminative re-ranking for spoken command understanding in HRI”, *in AI*AI*, 2013.

[11] T. J. Hazen, T. Burianek, J. Polifroni, and S. Seneff. “Recognition Confidence Scoring for Use in Speech Understanding Systems,” *Proc. ISCA tutorial and Research Workshop, ASR2000*, Paris, France. September 2000.

[12] C. Chelba, T. Mikolov, M. Schuster, Q. Ge, T. Brants, P. Koehn, T. Robinson. “One Billion Word Benchmark for Measuring Progress in Statistical Language Modeling”. eprint arXiv:1312.3005, 2013.

[13] K. Heafield. “KenLM: Faster and smaller language model queries,” *In Proceedings of the Sixth Workshop on Statistical Machine Translation*, ACL. pp 187–197, Edinburgh, Scotland, July. 2011.

[14] I. Androutsopoulos, and P. Malakasiotis. "A survey of paraphrasing and textual entailment methods." *Journal of Artificial Intelligence Research* (2010): 135-187.

[15] N. Madnani, and B. J. Dorr. "Generating phrasal and sentential paraphrases: A survey of data-driven methods." *Computational Linguistics* 36, no. 3 (2010): 341-387.

[16] G. A. Miller. “WordNet: A Lexical Database for English” *Communications of the ACM* Vol. 38, No. 11: 39-41, 1995.

[17] C. Fellbaum. “WordNet: An Electronic Lexical Database”. *Cambridge, MA: MIT Press*. 1998, ed.

[18] T. Mikolov, K. Chen, G, Corrado, and J. Dean. Efficient “Estimation of Word Representations in Vector Space”. *In Proceedings of Workshop at ICLR*, 2013.

[19] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. “Distributed Representations of Words and Phrases and their Compositionality”. *In Proceedings of NIPS*, 2013.

[20] J. Pennington, R. Socher, C. D. Manning. "GloVe: Global Vectors for Word Representation", *in Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)* 12 (2014).

[21] T. Joachims. “Optimizing search engines using click-through-data”. *In Proceedings of eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142, 2002.

[22] J. E. Hopcroft, and R. M. Karp. “An n^5/2 algorithm for maximum matchings in bipartite graphs,” *SIAM Journal on computing*, 2.4 (1973), pp. 225-231.Zhao, Shiqi, Xiang Lan, Ting Liu, and Sheng Li. "Application-driven statistical paraphrase generation." *In Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*: Volume 2-Volume 2, pp. 834-842. Association for Computational Linguistics, 2009.

[23] D. Kauchak, R. Barzilay. "Paraphrasing for automatic evaluation." *In Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pp. 455-462. Association for Computational Linguistics, 2006.