A Conceptual IR Chatbot Framework with Automated Keywords-based Vector Representation Generation

IOP Conf. Series: Materials Science and Engineering **769** (2020) 012020 doi:10.1088/1757-899X/769/1/012020

A Conceptual IR Chatbot Framework with Automated Keywords-based Vector Representation Generation

A S Lokman[1,2], **M A Ameedeen**[1,2] **and N A Ghani**[1,2]

[1] IBM Centre of Excellence, Cybercentre, Pahang Technology Park, Lebuhraya Tun Razak, 26300 Gambang, Kuantan, Pahang
[2] Faculty of Computing, Universiti Malaysia Pahang, Lebuhraya Tun Razak, 26300 Gambang, Kuantan, Pahang

abbas@ump.edu.my

Abstract. This paper proposes a conceptual remodel of Information Retrieval (IR) chatbot framework designed to eliminate the need for large Question-Answer (QA) pair dataset in chatbot's machine learning training and knowledge base development. Within ten proposed framework's components, we describe Ans2Q: a Neural Network model for question type approximation, and HR6: an IR score ranking calculation based on Ans2Q output. Fundamentally, these two components are the variance in which the proposed framework differs from others. Together with process flow explanation, we also provide several related formulas that hopefully can be used to implement this framework. Our general aim with this framework is to provide a tool that can be used to develop close domain chatbot with small knowledge and no readily available QA pair datasets.

1. Introduction

Development of Information Retrieval (IR) chatbot usually requires a large Question-Answer (QA) pair dataset. Current trends use Neural Network (NN) to predict the most similar Q to user's input query x, then retrieve the paired Answer (A) to the Question (Q) as chatbot's output/response. Major drawback for this model is the need for QA dataset. For general domain, there are lots of widely made available QA datasets [1][2][4], but for specific domains, such datasets might be unavailable due to small/non-specific domain scope [3]. As an example, new business venture might have all needed information to answer customer question regarding their product but do not yet have customer service conversational data. For matured organization, such conversational data is what is being used to create the QA pair dataset [5]. To create QA pair without Q data is a tedious process. For traditional rule-based chatbot, multiple keywords sets need to be created for each A response. For modern NN chatbot, Q for each A needs to be manually created by human. With motivation to provide non-tedious process of preparing QA pair for such datasets, we propose a new conceptual framework for IR chatbot that can automatically generate Q representation for each given A data. Our framework uses the traditional approach of keywords sets, while also implementing modern NN for question type classification and automated Keywords-Based Vector (K-VR) representation generation.

In essence, our paper makes following contributions:

- We propose a remodeled IR chatbot framework that is built upon K-VR similarity process.
- We propose a new score calculation method named HR6 for scoring IR candidate in ranking process based on weighted Kipling Method's word.
- We propose a new NN model for approximating question type based on sentences' parameters and its vector representation.

The rest of this paper is structured as follows: Next section will present related works that directly reflect various components in our proposed framework. Next subsequent section will discuss our propose framework from overall architecture to details on each components. This paper ends with future work in regards to our intended implementation and conclusion.

2. Related Work

Developing a chatbot with pre-developed framework has become a norm nowadays [6][7][8]. Although the framework design is mostly centered towards certain chatbot type, many argue that such design is needed in order to achieve full end-to-end usability in chatbot development. Modern chatbot design can generally be categorized into several base elements which are: 1) knowledge (open or close domain), 2) response generation (retrieval or generative), 3) text processing (vector embedding or latin alphabet), and 4) Machine Learning (ML) model (usually using neural network).

2.1. Knowledge Domain

A chatbot can be categorized as open or close domain. Domain in this sense is the knowledge that the chatbot is covering. Open domain means the chatbot must know general knowledge covering from recent news topic, entertainment, etc., as well as basic human knowledge. If the chatbot knowledge is specific to certain area such as customer service, psychology, etc., the system will be classified as close domain chatbot. Researchers have concluded that close domain chatbot is easier to build and currently producing good results [9][5][10] while open domain chatbot is still hard to build and have been producing a fair amount of false positive results [1][4].

2.2. Response Generation

Two basic methods for chatbots to generate/produce a response are retrieval and generative. A lot of variations from these two methods have emerged but the base process largely remains the same with retrieval being a process of selecting best output from shortlisted candidates, and generative being flexible output generation based on input sequence and trained classifiers. Among others, Seq2Seq [11] is the most famous model for both retrieval and generative methods. Originally designed for Neural Machine Translation (NMT) system, Seq2Seq architecture is based on encoder and decoder framework whereby encoder will receive input and decoder will produce output that is relevant to the input (like translating one language to another). Abbreviated from Sequence-to-Sequence phrase, Seq2Seq encoder takes input sentence as a sequence of words, analyzing each word (one-by-one in sequence) while generating its representation (one word at a time), and finally generating the whole input representation to be passed on to decoder module. Decoder then deciphers the input representation and generates an output (one word at a time) until all encoded representations are deciphered. With such capability and demonstrated good results in finding/calculating similarities (similarity of encoded representation between input and candidate output), researcher/developer has implemented Seq2Seq in ranking most similar output/response towards the input (retrieval) and also generate a totally flexible output in word or character based that is sequentially reflected the input (generative).

2.3. Text Processing

Word embedding (WE) or vector representation of word are real numbers in vector space that can denote a semantic relationship (by distributional hypothesis) between words within specific vocabulary. The relationship is defined based on distributional hypothesis that postulate "Words that occur in the same contexts tend to have similar meanings" [12]. Because it is a real number, statistical as well as arithmetical calculation towards WE is highly possible thus making it feasible to be used in statistical ML model. Concerning chatbot architectural design, most system that uses ML model (for training or real-time implementation) seem to also utilizes WE in their respective text processing modules.

2.4. Machine Learning Model

Machine learning nowadays is everywhere. Anything with big data repository is being transformed into ML powered systems, and any system with less data is being restructured to collect more data for future ML/AI roadmap. In modern chatbot architecture, ML (in particular Neural Network (NN)) can be seen as a core all-around technology where it is being used in input preparation and processing, as well as output processing and production.

The use of ML in chatbot's input preparation stage is mainly on generating WE. As discussed before, modern chatbot text processing module utilizes WE due to heavy use of statistical ML methods. Overall, there are two ways to generate vector representation of words which are count-method and predictive-method (ML method). While count-method is actually counting word co-occurrence based on specified context, predictive-method just predicts it. Although it seems unlikely (actual vs prediction), [13] systematic comparison has shown that predictive-method is far superior than count-method, thus becoming method of choice among researcher.

3. Framework

This paper proposes a remodel IR chatbot framework that is structured upon two main components: 1) Keywords-based vector representation (K-VR) and 2) Kipling representation (Kp). Both components will be auto-generated and become the representation of chatbot answer/response. In other words, K-VR and Kp is a Q pair for chatbot A data. For system to generate K-VR and Kp, A data must be segmented into sentence/s specific for answering one scope of question. For example, answer regarding phone number and address (contact information) should not be mixed with answer regarding product specification and type (product information). Following Figure 1 visualized our proposed framework in ten components within three interconnected layers.

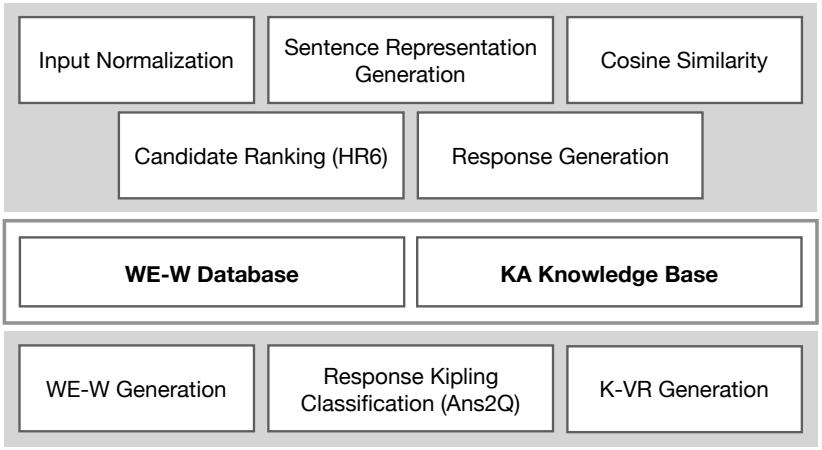

Figure 1. IR chatbot framework with K-VR and Kp (using Ans2Q) generation.

Referring to Figure 1, our proposed framework contains ten components grouped into three separate but interconnected layers. The top layer (Front-end layer) contains five components: 1) Input Normalization, 2) Sentence Representation Generation, 3) Cosine Similarity, 4) Candidate Ranking (HR6) and 5) Response Generation. The middle layer is a data layer that comprises of two databases: 1) WE-W Database and 2) KA Knowledge Base. The bottom layer (Back-end layer) contains three components: 1) WE-W Generation, 2) Response Kipling Classification (Ans2Q) and 3) K-VR Generation. Following are description for each layer:

Front-end layer - This layer will handle all real-time processes during chatbot interaction with user, from getting user's input until generating chatbot's output. Following Figure 2 depicted overall data flow for this front-end layer.

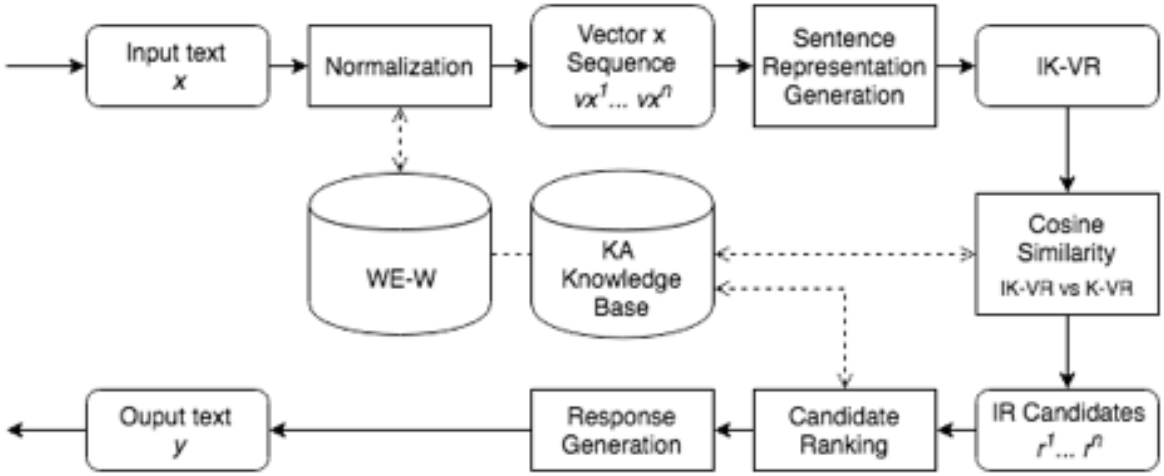

Figure 2. Data flow for front-end layer.

Referring to Figure 2, input x from user will go through Normalization process where each input word will be converted from latin alphabet to vector value based on WE-W (Word Embedding - Weight) database (WE-W will be further explained in framework component 6). Normalization will produce *Vector x Sequence* value that later be used in Sentence Representation Generation process (further explanation is in framework component 2). Next process is Cosine Similarity calculation between representation of user's query (IK-VR) and stored chatbot responses (K-VR). Further explanation on Cosine Similarity is in component 3 while K-VR is in component 10). Output from collective Cosine Similarity processes are top IR/response candidates that later will be scored using our proposed ranking function HR6 (further explanation is in component 4). Based on HR6 result, highest ranked answer will be generated by chatbot as an output y to user's input x.

Middle layer - This layer bridges a connection between front-end and back-end layer. Population of both databases in this layer is done through back-end layer components. After initial population, both databases will be ready for front-end layer utilization. Continuous KA Knowledge Base database population for new data will be done by back-end layer during implementation.

Back-end layer - This layer will handle all pre-processing details before front-end layer can utilizes the middle layer. After going live (implementation), this layer will be used for updating middle layer with new data (if any). In usual circumstances, only Ans2Q and K-VR Generation components will be used during implementation phase.

As briefly described above, connection between all three layers are through the middle layer such that front-end and back-end layer are not directly connected. With regards to given description, next sub sections will explain in detail each component functionalities together with its underlying logic calculation.

3.1. Input Normalization

This first component of front-end layer will take user's raw text data as input x, tokenize it into each individual word/symbol, retrieve each token's embedding from WE-W database, and create the vector x sequence $(vx^1 \dots vx^n)$ where vx^i is the vector embedding and weight for each respective token. Token that did not have its embedding data in WE-W will be represent as zero vector $(0^{\rightarrow})$ with zero weight value.

3.2. Sentence Representation Generation

This second component of front-end layer will take vx sequence and produce IK-VR (*Input Keywords - Vector Representation*) value for every vx weight that is more than threshold t. Formula for IK-VR calculation is as follows:

$$\text{IKVR} = \frac{1}{n}\sum_{i=1}^{n}(\text{WE} \cdot \text{tfidf})_i \text{ for tfidf} > t \quad (1)$$

3.3. Cosine Similarity

This third component of front-end layer will take IK-VR value and compare it with K-VR value from KA Knowledge Base using cosine similarity measurement. Calculation formula is as follows:

$$\cos(\theta) = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \quad (2)$$

Given equation calculates the cosine similarity between two vectors where A_i and B_i are components of IK-VR and K-VR respectively. Output from this component is top ten list of IR candidates $(r^1 \dots r^{10})$ that have the highest cosine similarity score.

3.4. Candidate Ranking (HR6)

The forth component of front-end layer will ranked IR candidates by calculating its HR6 score. HR6 (abbreviated Highest Ranking Six) is proposed to substitute the usual IR ranking function such as Okapi BM25 [14] by incorporating Kipling representation value Kp produced by our Ans2Q model. To generalize our function, Kipling representation Softmax KpS will be used instead of Kp one-hot representation. While one-hot representation is needed for supervised Ans2Q training (as discrete label class), HR6 needed more flexible value to cater for possible appearance of all six Kipling question words (Who, What, When, Where, Why and How) in user's input query. As such, we propose HR6 to be calculated using following equation (3). Given IK-VR from user input query Q and KpS value linked to candidate answer R from KA Knowledge Base, the HR6 score of R is:

$$\text{score}(R, Q, KpS) = -\cos(\theta) + \left(\overrightarrow{KpS} \cdot \overrightarrow{IKp}\right), \quad (3)$$

where $\cos(\theta)$ equals cosine similarity between IK-VR and K-VR as calculated in *Cosine Similarity* equation, $KpS \rightarrow$ equals vector value of KpS (Kipling representation Softmax) associated with candidate *R* and $IKp \rightarrow$ equals vector value of IKp (Input Kipling representation) associated with input query *Q*. *KpS* is a six-dimensional vector with sum to one value, generated from Ans2Q Softmax layer. *KpS* is an original Softmax value before being transformed into one-hot encoding of *Kp* value (further explanation is in section 3.9. Response Kipling Clasification (Ans2Q)). *IKp* is a six-dimensional one-hot encoding value generated by matching each word in input query to six Kipling question words. We defined the sequence of Kipling words to be as follows: Who, What, When, Where, Why and How. As such, if the word "When" is presents in input query, *IKp* value would be [0,0,1,0,0,0]. We expect the input query to contain no more than one Kipling word but in the case where multiple words are presence, HR6 formula should relatively work the same as it calculates the dot product of two vectors (*KpS* and *IKp*). In essence, candidate R will get higher HR6 score when: 1) Cosine similarity value is low, 2) *IKp* has one *true* value (not all *false* or *zero* value), and 3) *IKp true* value index is same with *KpS* highest value index.

After all HR6 scores for IR candidates have been calculated, this component will rank candidates ($r^1 \dots r^{10}$) with highest HR6 score on top to lowest HR6 score on bottom (highest score means the best matching candidate).

3.5. Response Generation

The fifth or last component of front-end layer will generate response (output y) back to user based on candidate ranking result. The highest ranked answer *A* linked to candidate K-VR and *Kp* value will be generated as output y provided that its HR6 score is above certain threshold. If the highest HR6 score is below the set threshold, this component will classify user's query as not-understandable by chatbot thus generating generic response as to get more/new input data.

3.6. WE-W Database

The first component of middle layer is Word Embedding - Weight (WE-W) database. WE-W contains two separate values for every word in general and specific dictionary. Those values are Word Embedding *WE* and Weight *W*. To populate WE-W as to become cumulative embedding database, two sets of dictionaries are required. One is general dictionary (base embedding) and the other is specific dictionary (complement embedding). General dictionary can be from any available public corpora (depending on chosen language), while specific dictionary must be from intended close domain corpora. Regarding this framework, WE-W database will be populated using component *8. WE-W Generation*.

3.7. KA Knowledge Base

The second component of middle layer is Keywords-Answer (KA) Knowledge Base. Generally, this database contains prewritten response/answer for chatbot to generate. Specifically, this database comprises of three separate values for every response item. Those values are: 1) *K* or K-VR that is a single representation of keywords for each *A*, generated using component *10. K-VR Generation* of this framework, 2) *A* that is a chatbot response/answer directly retrieved from specific/close-domain document, and 3) *Kp* that is a Kipling representation generated using Ans2Q model (component *9. Response Kipling Classification (Ans2Q)* of this framework). To get *K-VR* and *Kp*, WE-W must first be produced. Regarding this framework, K will be generated using component 10. K-VR Generation while Kp will be generated using component 9. Response Kipling Classification (Ans2Q).

3.8. WE-W Generation

The first component of back-end layer is a module to generate WE-W value for component 6. WE-W database. As mention before, WE-W is a group of two separate values which are WE and W. To generate WE, one can use any WE methods available nowadays such as GloVe [15] and Word2Vec [16]. Same for W, any word weighting methods can be used to calculate W. One of most used method in IR and chatbot field is TF-IDF. Term Frequency – Inverse Document Frequency (TF-IDF) is a numerical statistical method (a product of two statistics: TF and IDF) that calculates the importance of term (a word or phrase) based on its occurrence frequency in multiple documents (or in defined corpus space). The TF-IDF value increases proportionally to the number of times a term appears in the document (TF) [17] and is counterbalance by the number of times a term appears in the whole corpus (IDF) [18]. In other word, TF-IDF value is high when the term appears a lot in certain documents, but is low when the term appears a lot in a whole dictionary.

3.9. Response Kipling Classification (Ans2Q)

The second component of back-end layer is a module to generate *Kp* value for component *7. KA Knowledge Base*. To generate Kp, we propose a new NN model named Ans2Q (abbr. Answer-to-Question). To put thing into perspective as to why this model is proposed, consider following two questions: "How to do mileage claim?" and "What is mileage claim?". With IK-VR generation, every input word will be assigned a weight value that represents its semantic significance towards the whole

dictionary corpus. As such, the words "how", "to", "do", "what" and "is" will be regard as insignificance while the words "mileage" and "claim" will become the input's keywords. For that reason, both questions will have same interpretation by the system while in true context its represent different question type that requires different answer. Ans2Q intent to solve this issue by creating another tier in Answer representation that is a probability of which question type is most likely to be use in order to get the Answer. Following Figure 3 illustrates our proposed neural network model.

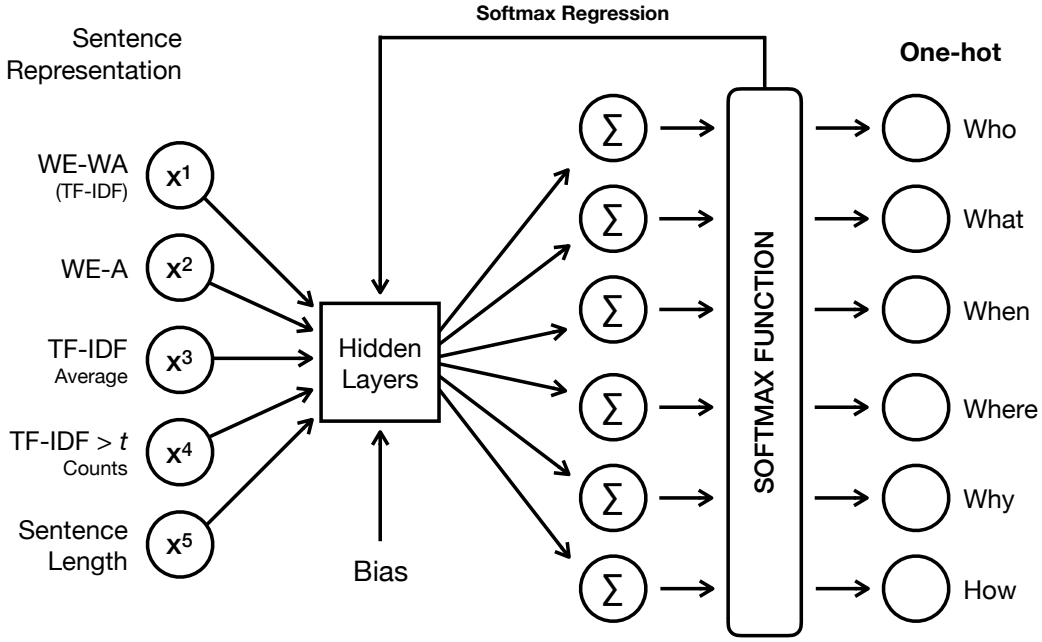

Figure 3. Neural network model for multinomial classification of Kipling question type.

Proposed model is intended to predict one-hot value of Kipling representation *Kp* for each Answer *A* data based on several independent variables of sentence/Answer representation. One-hot value is produced from Softmax function that takes the final value from Softmax Regression process that is done towards the model's hidden layers. Independent variables (input features) for Ans2Q are as follows:

- WE-WA (TF-IDF): Word Embedding - Weighted Average using TF-IDF value as weight for all word in the sentence.
- WE-A: Word Embedding - Average of all word in the sentence.
- TF-IDF Average: Average value of all words' TF-IDF.
- TF-IDF > t Counts: Total number of word with TF-IDF more than threshold.
- Sentence Length: Total number of word/token in the sentence.

All independent variables are Real Number value that need to pre-calculate prior to entering this model. As for dependent variables of one-hot value (label classification), it is a six-dimensional vector with binary matrix value (e.g [1,0,0,0,0,0] for Who and [0,1,0,0,0,0] for What). This vector will be determined based on Softmax function output where highest probability will be valued 1, while others will be 0 (this is the Kp value). Despite one-hot value, all original Softmax values for all classes will also be stored in database for later used (this is a complimented Kp value denoted as Kipling representation Softmax or KpS). In general, our proposed multinomial distribution probability can be

represented using following Softmax equation that is an extension of logistic regression formula (sigmoid function) towards multiple classes problem.

$$p(y = j|\mathrm{x}) = \frac{e^{(\mathrm{w}_j^T \mathrm{x} + b_j)}}{\sum_{k \in K} e^{(\mathrm{w}_k^T \mathrm{x} + b_k)}} \quad (4)$$

Given Softmax equation calculates the probability of *Kp* class with index *j* based on given value x where w is weight, x is feature value, *b* is bias and *k* is the class label. *T* in the formula refers to applied layer (the Softmax Function layer) where in our case is represented by (6,1) vector; a six-dimensional vector with sum to one value.

3.10. K-VR Generation

The third component of back-end layer is a module to generate K-VR value for component *7. KA Knowledge Base*. K value is a vector representation (in real number) of keywords, designate for each *A* data in KA Knowledge Base. In our work, we denote *K* as K-VR that is an abbreviation of *Keywords - Vector Representation*. K-VR is calculate using following equation where *n* equals total number of word in *A* data with TF-IDF value is more than *t* threshold. The implementation of *t* is to limit *K* value as to only represents significant word for each document in specific dictionary, hence the term "keywords" (to be noted that K-VR equation is the same as IK-VR equation described in section 3.2. Sentence Representation Generation).

$$\mathrm{KVR} = \frac{1}{n} \sum_{i=1}^{n} (\mathrm{WE} \cdot \mathrm{tfidf})_i \text{ for } \mathrm{tfidf} > t \quad (5)$$

4. Future Work

As with other conceptual proposal, proof of concept implementation is crucial. For this proposed framework, first task is to perform data preparation for middle layer database storage. We intend to pursue this by using specific banking institution (Bank X) Standard Operating Procedure (SOP) as our small scope domain data. Our case study is to develop a chatbot that can answer Bank X staffs' questions regarding their organizational SOP. Raw SOP documents will be normalized as to become group of sentence that represent specific topic in question. With normalized SOP data, back-end layer components can be prepared and trained to generate needed QA pairs for front-end layer components and its real-time processes.

5. Conclusion

This paper proposed a conceptual framework for IR chatbot that focuses on automated Q (question) representation generation. Our framework's main contributions are Ans2Q; a NN model for question type approximation and HR6; an IR score ranking calculation based on Ans2Q output. Although only conceptual, we provided detailed explanation on functionalities as well as calculation formula such that it is clear for anyone to implement this framework. Same with other NN training, biggest implementation challenge would be to fine tune Ans2Q so that Kp approximation is as good as manual human process. Sentence/answer representation parameters/features that goes into Ans2Q also need to be reevaluated after training as those proposed are only as an initial hypothesis.

References

[1] Serban, I. V., Sankar, C., Germain, M., Zhang, S., Lin, Z., Subramanian, S., Kim, T., Pieper, M., Chandar, S., Ke, N. R., Rajeshwar, S., Brebisson, A., Sotelo, J. M. R., Suhubdy, D., Michalski, V., Nguyen, A., Pineau, J., Bengio, Y. (2017). A deep reinforcement learning chatbot. *arXiv preprint arXiv:1709.02349*.

[2] Lokman, A. S., & Ameedeen, M. A. (2018, November). Modern Chatbot Systems: A Technical Review. In *Proceedings of the Future Technologies Conference* (pp. 1012-1023). Springer, Cham.

[3] Lokman, A. S., & Jasni M. Z. (2010). Chatbot enhanced algorithms: a case study on implementation in Bahasa Malaysia human language. In *International Conference on Networked Digital Technologies*, pp. 31-44. Springer, Berlin, Heidelberg.
[4] Liu, H., Lin, T., Sun, H., Lin, W., Chang, C. W., Zhong, T., & Rudnicky, A. (2017). RubyStar: A Non-Task-Oriented Mixture Model Dialog System. *arXiv preprint arXiv:1711.02781*.
[5] Qiu, M., Li, F. L., Wang, S., Gao, X., Chen, Y., Zhao, W., Chen, H., Huang, J., Chu, W. (2017). Alime chat: A sequence to sequence and rerank based chatbot engine. *In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, (Vol. 2, pp. 498-503).
[6] Wei, C., Yu, Z., & Fong, S. (2018, February). How to Build a Chatbot: Chatbot Framework and its Capabilities. In *Proceedings of the 2018 10th International Conference on Machine Learning and Computing* (pp. 369-373). ACM.
[7] Lommatzsch, A. (2018, June). A Next Generation Chatbot-Framework for the Public Administration. In *International Conference on Innovations for Community Services* (pp. 127-141). Springer, Cham.
[8] Ma, S. P., & Ho, C. T. (2018). Modularized and Flow-Based Approach to Chatbot Design and Deployment. *Journal of Information Science & Engineering*, *34*(5).
[9] Yin, Z., Chang, K. H., & Zhang, R. (2017, August). DeepProbe: Information Directed Sequence Understanding and Chatbot Design via Recurrent Neural Networks. *In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 2131-2139). ACM.
[10] Cui, L., Huang, S., Wei, F., Tan, C., Duan, C., & Zhou, M. (2017). SuperAgent: A Customer Service Chatbot for E-commerce Websites. *Proceedings of ACL 2017, System Demonstrations*, 97-102.
[11] Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. *In Advances in neural information processing systems* (pp. 3104-3112).
[12] Harris, Z. S. (1954). Distributional structure. *Word*, *10*(2-3), 146-162.
[13] Baroni, M., Dinu, G., & Kruszewski, G. (2014). Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (Vol. 1, pp. 238-247).
[14] Robertson, S., & Zaragoza, H. (2009). The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4), 333-389.
[15] Pennington, J., Socher, R., & Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (pp. 1532-1543).
[16] Mikolov T., Chen K., Corrado G., & Dean J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.
[17] Luhn, H. P. (1957). 1 A Statistical Approach to Mechanized Encoding. *IBM journal*.
[18] Sparck Jones, K. (1972). A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, *28*(1), 11-21.

Acknowledgments
This work was supported in part by Department of Higher Education, Ministry of Education Malaysia under the Fundamental Research Grant Scheme (FRGS), through Universiti Malaysia Pahang (Ref: FRGS/1/2018/ICT02/UMP/02/12).