ASIAN JOURNAL OF PHARMACEUTICAL AND CLINICAL RESEARCH

INNOVARE ACADEMIC SCIENCES
Knowledge to Innovation

Special Issue (April)

Online - 2455-3891
Print - 0974-2441

Full Proceeding Paper

WORKING OF A CONTEXT-AWARE CONVERSATIONAL ENTITY

PRAGYA SHRIVASTAVA, BHARADWAJA KUMAR G*

School of Computing Science and Engineering, Vellore Institute of Technology University, Chennai Campus, Chennai, Tamil Nadu, India.
Email: bharadwaja.kumar@vit.ac.in

Received: 19 January 2017, Revised and Accepted: 20 February 2017

ABSTRACT

Introduction of new technologies into the world is increasing rapidly, and to assist the users to get equipped with such technologies, industries are providing customer care services. Contacting a customer care service is subjective to several overheads of selecting options from a listed set, waiting for the switching between selections, and awaiting the support of a customer care executive as the process usually requires a human intervention. Hence, a substitute for personnel is required by the IT industries to automate the communication process in assisting the customers. Chatbots with context aware question-answering capabilities can be viewed as a good solution to such customer care assistance. Development of a chatbot and the complexities involved in getting it to work effectively is delineated in this paper.

Keywords: ncreasing rapidly, and to assist the users

 DOI: http://dx.doi.org/10.22159/ajpcr.2017.v10s1.19638

INTRODUCTION

An efficient way of communication between human beings is with the help of a "language." Therefore, making the machine to understand our way of communication is always a very essential point of research. The computer cannot understand the language that we human beings speak as such; we need to convert this human-understandable language into a machine-understandable form. The actual objective is to make the machine understand the significance of a language's grammar, phrases, idioms, vocabulary, and also the context so that the machine can respond to questions put forward to it in a way similar to us. An example of a chatbot's working [1] is given in Fig. 1. To achieve this objective, many applications such as inverted indexing, cosine similarity, and artificial intelligence markup language [2] have been developed, which is primarily supported by the bags of words approach and hence lacks in semantics.

Automating the text analysis by a machine is not an easy task; it involves a lot of challenges in providing a deep understanding of natural language to machines. Some very successful solutions given by IT industries to assist people are Apple's SIRI and IBM's Watson. Although both these systems are successful in the aspect of providing user assistance, there exist some limitations on its computational requirements and its performance. The technical processes involved in making the natural language understandable to the machine include preprocessing, segment splitting, intention identification, relevant answer retrieval, and then framing the answer to the user. This understanding, interpretation, and extraction of meaningful sentences to develop a conversational entity are performed with the help of natural language processing (NLP) [3].

The two main entities of an NLP domain in developing any conversational agent include language understanding (U) and language generation (G) and are depicted in Fig. 2. NLP is a domain which enables us in the development of diversified applications such as:

1. Grammar checkers
2. Question-answering systems
3. Information extractors
4. Machine translation and interpreters.

This paper focuses on delineating the working of a conversational entity, in other words a question-answering system dealing with a particular context. Any question-answering system can be developed by making the machine to learn on which question should be mapped with which answer, and mapping of question answer can also be viewed as a sequence to sequence mapping task. Recurrent neural networks (RNNs) [4] with encoder and decoder have proved to be successful for sequence-to-sequence mapping [5] since it reduces the manual task while training the model. The sequence-to-sequence mapping task includes machine translation, named entity recognition, etc. [4]. The neural network (NN) of sequence-to-sequence model is jointly trained to maximize the conditional probability of mapping the correct answer on getting the context aware question or any related question [6].

LITERATURE SURVEY

This section details the three main NLP tasks which are included in the architecture of any conversational model: Part-of-speech (POS) tagger, text similarity, and encoder-decoder framework.

POS tagging

Knowledge on the POS of a sentence of a language is essential to understand, segment, or check the sentences given to the system. In English language, totally, eight POS [7] are available and are given as follows: Noun, pronoun, adjective, verb, adverb, article, participles, and auxiliaries. Prediction of the possible occurrence of the next word from an existing entry of a word can also make possible with the help of information about POS in a sentence. For example, after an occurrence of an article, the probability of appearance of a noun is 91%. Hence, generally, after using an article, we will use a noun. For words where the meaning is ambiguous but the POS has been identified correctly, the system can try to rectify and place the word with the intended meaning if some spelling errors or contextual errors have occurred. Generally, POS tagging application is developed by training the model with the large corpus and then by applying probability distribution on them. After that, the model gets ready to test its POS tagging capabilities on any unseen text.

Text similarity

Natural language is ambiguous in nature as the same concepts or statements can be expressed in different ways and some different meanings can be conveyed with the usage of same phrases but in different context. Interpretation of natural language when such cases are present in the text becomes complex for the machines to decode. To get a better understanding of natural language, few algorithms

have been introduced in the past which help to establish the correct intention being conveyed and to find the key point (attention) of the user's query or question. Similarities in the text can be listed into three categories [8] and are listed below:

- Morphological similarity
- Spelling similarity
- Semantic similarity.

Morphological similarity

In English, there are different forms of words and a slight change in the word affects the meaning, usage, tense, as well as the POS of the particular word. Example: Work, works, worked, worker, working are all forms of the same word with different tenses and POS. However, all forms of that word will always indicate a similar meaning, i.e., the base meaning will never change.

Storing each and every possible word in a language and along with its various word formations is a highly impossible task in any language. Hence, to reduce the length of the vocabulary, only the base/root form of every word in the language is stored. A vocabulary holds most of the words in the language, and not all the words, storing all the words is never possible for any language. To obtain the root/base form of every word, potters stemming method is used and some examples are given in Figs. 3-5. It stems down each word to its base form so that every form of a word can be mapped to one. By this process, the base meaning of the word is preserved which helps in understanding the keyword around which the whole statement is framed. It also helps in breaking down the complex sentence into simpler smaller segments. Sometimes, this process might distort the intention of the statement, but it is very useful when we have a large corpus with a huge number of words and we cannot store every form of word in the vocabulary [9]. For example, the words "compute" and "computer" become "comput" after stemming. With the help of this root word, the machine will understand that context is related as both the words are formed from the same root word.

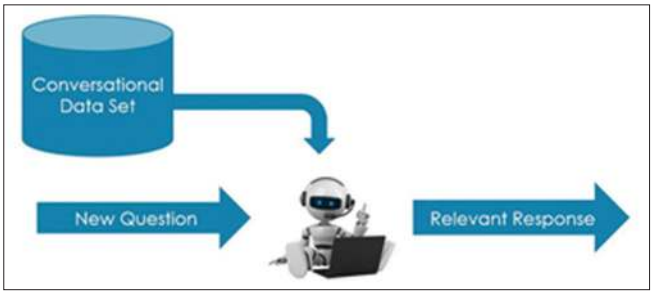

Fig. 1: Chatbot

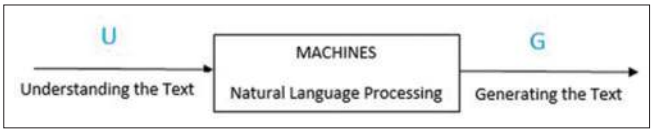

Fig. 2: Natural language processing

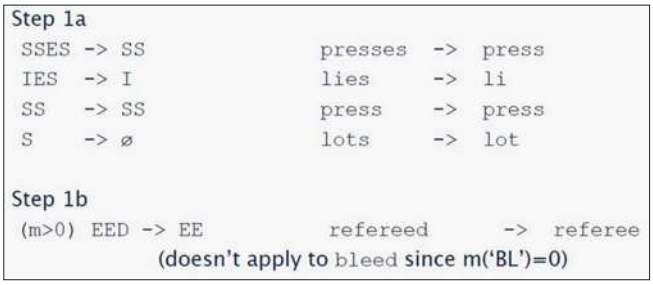

Fig. 3: Stemming ruleset-1 as proposed by Potter's method

Spelling similarity

When we intend to write down a particular word due to a spelling mistake and sometimes the similarity which exists between the intended word and some other word, the word turns out to be a totally another one. For example: "Leather" can be mistakenly typed into "Weather." Spelling mistake could be result of anything such as typing error, change in the pronunciation, or lacking knowledge of any word and many more. Hence, intelligence is required by the machine to either assume the correct replacement or to rectify the spelling error. Sometimes, replacing the words deals with the context on which previous conversation is done or sometimes on the particular topic on which the bot has been developed. The replacement of a word due to similarity and spelling mistake can be dealt by addressing the number of conversions required to change the current word into the actually intended word. For doing this conversion, we need to have a count on the number of characters which will get inserted/deleted/substituted with appropriate costs for each conversion. By summing the cost of all these individual conversions in a word, we get the total cost to convert one word into another and then the word with least cost is preferred.

To apply this algorithm, first, we check if the word is present in the vocabulary or not. If not, then we apply edit distance algorithm that will calculate the cost of converting the given word into any other word from the vocabulary. Then, the word which has least cost of conversion will replace the given word and working of the edit distance algorithm is given in Fig. 6.

Semantic similarity

Semantic similarity describes the set of words which point to the same meaning. Sometimes, the same word acts differently depending on its position. Hence, to first understand this, POS tagging is needed, and then on the basis of the tag of that word, its synonyms are extracted. For example, "address" can be treated as a noun as well as a verb depending upon its position in the statement [9].

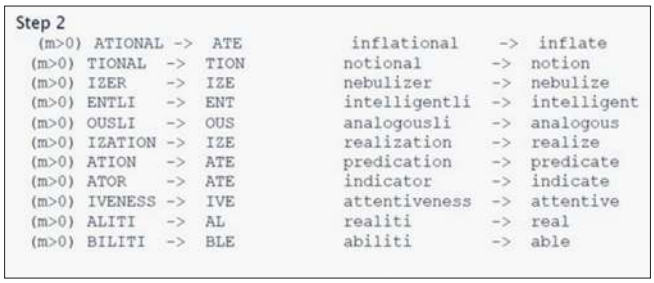

Fig. 4: Stemming ruleset-2 as proposed by Potter's method

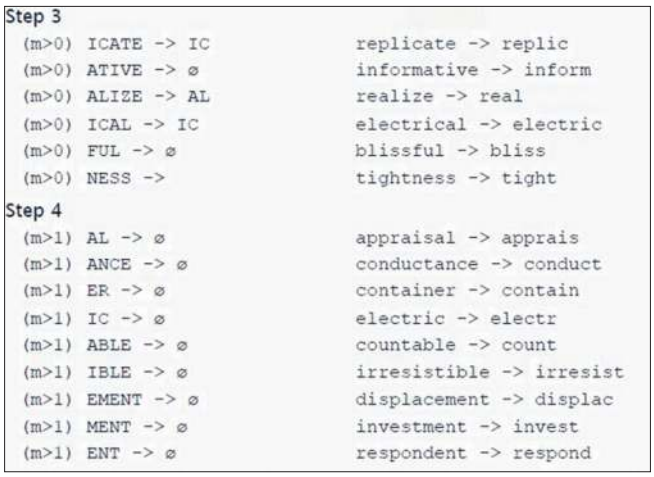

Fig. 5: Stemming ruleset-3 as proposed by Potter's method

For estimating the semantic similarity, Dekang Lin algorithm which is built on the basis of a wordnet is used. This algorithm is represented in a tree structure and stores the relation between hypernymy and hyponymy of each word. To check the similarity between two words, it estimates the probability of lowest common subsummer of the word under analysis and the intended word. Using this probability values, we can estimate how much closer these words are to each other. The formula used in the Dekang Lin algorithm to calculate the subsum is given in Fig. 7, and the tree structure followed by the wordnet is depicted in Fig. 8.

RNN-based sequence-to-sequence model

Artificial NNs (ANNs) are inspired from the information processing capabilities of our biological brains. We have many NN [10] architectures such as feed forward NN (FFNN), RNNs [11], and long short-term memory (LSTMs). RNNs are the ANNs with cyclical connections, and an RNN [12] can actually map the sequence of entire length of previous input sequences to each output layer.

Sequence-to-sequence mapping tasks [13] include machine translation, question-answering systems, and conversation agents. However, in case of dialog process, multiple turns of inputs are needed unlike other sequence-to-sequence mapping process. Deep NNs (DNNs) has achieved an excellent performance on many tasks such as image recognition, speech recognition, and many more. However, DNNs cannot be applied to problems where length of the sequences is unknown *a-priori*, for example, machine translation task. RNN-based approaches have proved to be successful in sequence-to-sequence mapping task.

RNNs can be thought of FFNN unfolded in time space, and hence, they suffer from the problem of what is known as exploding and vanishing gradients problem in the literature. LSTM architecture is designed to overcome the problem of vanishing gradients through gating mechanism. LSTM usually consists of memory cells and all those cells have three multiplicative units - the input gates, the output gates, and the forget gates that allow us to read, write, and reset the cells. LSTMs have been proven to be successful in learning long-term temporal dependencies, and hence, they can be effectively used in mapping sequences with even longer input and output sequences. Several attempts have been carried out to address the sequence-to-sequence learning with the support of NNs where encoder and decoder have been utilized. Since we want to build the conversational agent, our aim is to predict the response (the next sentence), given the context (previous sentences in the conversation). Hence, we aim to develop a context-specific chatbot and we wish to reduce the regular problem of dealing with a large vocabulary. Our approach is based on sequence-to-sequence model and is described in the study done by Buck *et al.* [16].

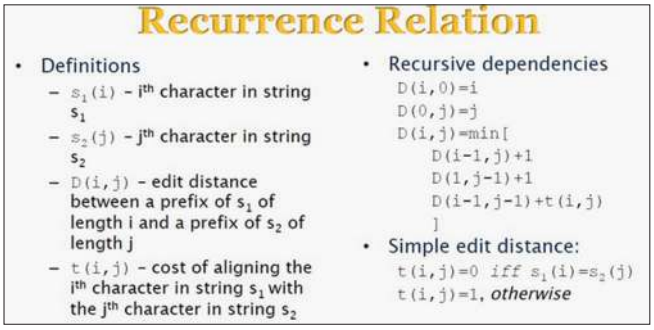

Fig. 6: Edit distance rules

$$IC(c) = -\log P(c)$$

$$Sim(v,w) = 2 \times \log P(LCS(v,w)) / (\log P(v) + \log P(w))$$

Fig. 7: Formula of Dekang Lin

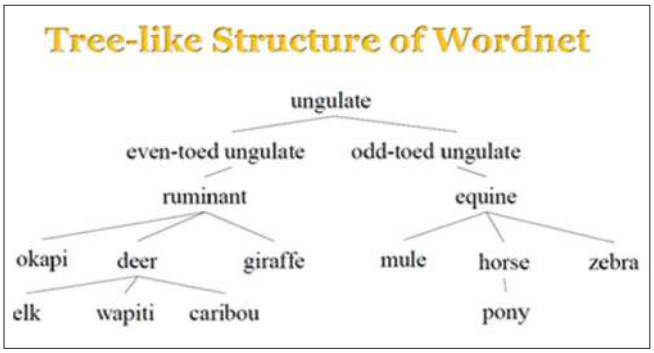

Fig. 8: Example of wordnet

ARCHITECTURE

The development of a context-specific bot is our prime concern, and this bot should be aware of contextual similarities and differences to mimic the conversation on a topic with the human user. To develop such an application, we do not need to manage large vocabulary because our context will always be limited, but we need to include an architecture that can easily evaluate the question as well as can maintain a track of even long conversations. The architecture that is generally used to accomplish this goal is depicted in Fig. 9 and consists of two major modules: Query modulation and question-answer mapping.

Query modulation

Whenever a user asks the question, it is passed through some filters to convert the question into a modulated query. Three filters are used to convert the question into a query: Spelling correction, stemming, and synonyms finder. First, the user's question is tokenized and sent for spelling correction, and after that, each and every word will be passed through stemmer which will step-down the words into its basic form so that machine can categorize the word into an existing word in the vocabulary or into a new word. At the end, the words will be passed through a synonym finder to list all the possible synonyms of each word so that even if a known question is asked in a slightly different way, the machine can find its way to answer the question. At its final stage, the question is reconstructed by replacing every possible word with its synonym from the vocabulary and this reconstructed query will be given as an input to the next level.

Question-answer mapping

The query, received from the first module, will be tokenized, and then, it will be given word-by-word to the NN. This approach uses an RNN model where the input sequence is read as one word at a time to obtain a fixed length vector representation termed as thought vector and another RNN is used to generate the output sequence which is conditioned on the vector obtained from the first RNN. Fig. 10 shows the processing of a query using sequence-to-sequence mapping in an RNN.

This sequence-to-sequence mapping model can be used for question/answering and conversations with very little changes in the architecture. Here, the input sequence is the context (series of previous conversation) and the output sequence is a reply. During the training, the true output is fed into the decoder and learning takes place using back propagation. The model is trained to maximize the probability of the true reply given the source sentence. During the inference/testing, we feed the predicted word at the current time step as the input in the next step.

IMPLEMENTATION

A python-based stemmer, spell checker, and synonyms finder [14] has been utilized and tensor flow has been used for applying the RNN to establish the sequence-to-sequence model.

Spell checker

To check the spellings of words, a vocabulary has been created which consists of all the words that could be used for the conversation.

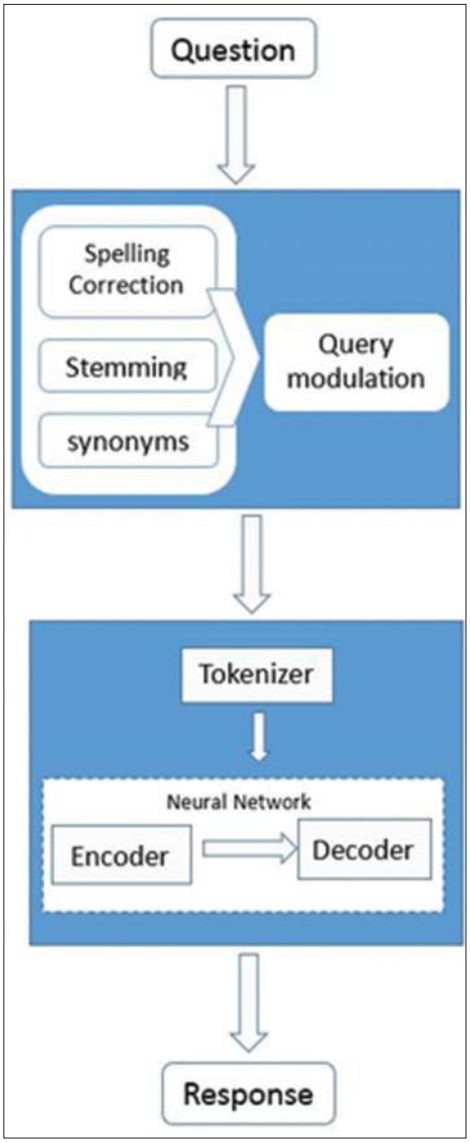

Fig. 9: Architecture

Whenever user enters the word, the program first checks whether the word exists in the vocabulary or not. If not, then it takes each possible word from the vocabulary and calculates the number of changes required to convert one word into another. Taking the word with the smallest distance, it checks whether the distance is less than the threshold value or not. If it is less than that the threshold, it replaces the word or else it keeps the word as it is.

Synonyms finder

A complete sentence is taken as input and POS tagging is performed according to the position where it is placed in the sentence. After this, it finds a list of possible synonyms for the word according to its POS. At the end, it checks which word exists in the vocabulary, and according to that, it alters the sentence [15]. For all those words whose synonym does not exist or whose synonyms are not in the vocabulary are left unchanged.

CONCLUSION

Achieving a good replacement to human assistance is not so easy, and developing human-computer interfaces is very difficult owing to the ambiguities present in NLP. Some of the challenges in achieving a chatbot have been discussed in this paper. Although a lot of research work has already been carried out, a lot more needs to be done for developing such intelligent and interactive systems.

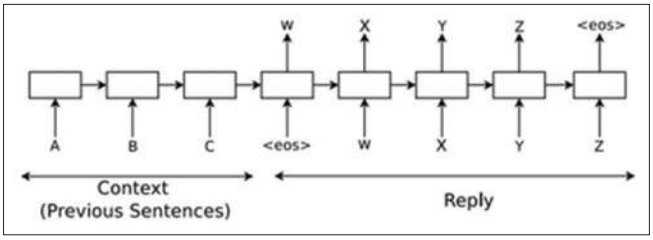

Fig. 10: Using the sequence to sequence framework for modeling conversations

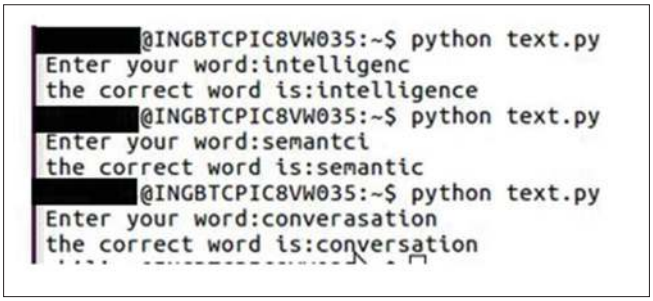

Fig. 11: Snapshot of explicit output of spelling checker

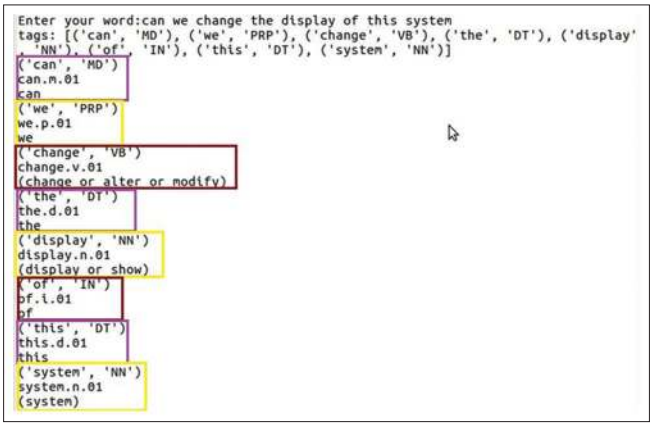

Fig. 12: Snapshot of explicit output of spelling checker

REFERENCES

1. Abdul-Kader SA, Woods J. Survey on chat bot design techniques in speech conversation systems. Int J Adv Comput Sci Appl (IJACSA) 2015;6(7):1.
2. Marietto MD, de Aguiar RV, Barbosa GD, Botelho WT, Pimentel E, França RD, *et al*. Artificial intelligence markup language: A brief tutorial. Int J Comput Sci Eng Surv 2013;4(3):1-20.
3. Chowdhury GG. Natural language processing. Annu Rev Inf Sci Technol 2003;37(1):51-89.
4. Yao K, Zweig G, Peng B. Attention with intention for a neural network conversation model. 29 October, 2015. arXiv preprint arXiv:1510.08565.
5. Bahdanau D, Cho K, Bengio Y. Neural Machine Translation by Jointly Learning to Align and Translate. In: Proceedings of the International Conference on Learning Representations (ICLR), San Diego, CA; 2015.
6. Zhou X, Hu B, Chen Q, Tang B, Wang X. Answer sequence learning with neural networks for answer selection in community question answering. 22 June, 2015. arXiv preprint arXiv:1506.06490.
7. Taylor A, Marcus M, Santorini B. The Penn treebank: An overview. In: Treebanks. Netherlands. Springer; 2003. p. 5-22.
8. Steven B, Klein E, Loper E. Natural Language Processing with Python. Sebastopol, CA: OReilly Media Inc.; 2009.
9. Zou WY, Socher R, Cer DM, Manning CD. Bilingual word embeddings for phrase-based machine translation. In: EMNLP. October, 2013.

p. 1393-8.
10. Bengio Y, Ducharme R, Vincent P, Jauvin C. A neural probabilistic language model. J Mach Learn Res 2003;3:1137-55.
11. Vinyals O, Le Q. A neural conversational model. 19 June, 2015. arXiv preprint arXiv:1506.05869.
12. Sutskever I, Vinyals O, Le QV. Sequence to sequence learning with neural networks. In: Advances in Neural Information Processing Systems. Cambridge: MIT Press; 2014. p. 3104-12.
13. Graves A. Supervised sequence labelling. In: Supervised Sequence Labelling with Recurrent Neural Networks. Berlin, Heidelberg: Springer; 2012. p. 5-13.
14. Bird S. NLTK: The Natural Language Toolkit. In: Proceedings of the COLING/ACL on Interactive Presentation Sessions 2006 July 17. Association for Computational Linguistics. p. 69-72.
15. Lowe R, Pow N, Serban I, Pineau J. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. 30 June, 2015. arXiv preprint arXiv:1506.08909.
16. Buck C, Heafield K, van Ooyen B. N-gram counts and language models from the common crawl. In: LREC. Vol. 2. May, 2014. p. 4.
17. Jean S, Cho K, Memisevic R, Bengio Y. On using very large target vocabulary for neural machine translation. 18 March, 2015. erforarXiv:1412.2007v2.