Distributed Storage and Recall of Sentences

Marc Ebner
Ernst Moritz Arndt Universität Greifswald, Institut für Mathematik und Informatik
Walther-Rathenau-Str. 47, 17487 Greifswald, Germany
marc.ebner@uni-greifswald.de

***Abstract*—The human brain is able to learn language by processing written or spoken language. Recently, several deep neural networks have been successfully used for natural language generation. Even though it is possible to train such networks, it remains unknown how these networks (or the brain) actually processes language. A scalable method for distributed storage and recall of sentences within a neural network is presented. A corpus of 59 million words was used for training. A system using this method is able to find sentences which can be considered reasonable replies to an input sentence. The system first selects a small number of seeds words which occur with low frequency in the corpus. These seed words are then used to generate answer sentences. Possible answers are scored using statistical data also obtained from the corpus. A number of sample answers generated by the system are shown to illustrate how the method works.**

***Index Terms*—Natural language generation, neural networks, statistical natural language processing, artificial intelligence**

I. Introduction

The human brain consists of approximately 10^{11} interconnected neurons forming a large scale neural network. Using this network, humans are able to produce language and to understand written or spoken language. Presumably, many animals are also able to communicate with each other. The question is, how does natural language processing work in the brain? With this contribution, we try to answer this question.

First, we will discuss different theories on how language generation may be achieved by the brain. Unfortunately, these methods do not scale well to handle long sentences. Because of this, they are unlikely to be used by the brain. The work presented here, will show how to transform a large text corpus into fragments which are stored locally. These fragments are subsequently used for sentence generation. The system is able to generate seemingly meaningful replies to input sentences which are fed to the system. Instead of fully training the system on a large corpus and then generating replies, it is also possible to iteratively improve the system by taking every input as learning input. In this case, generated output will improve as more and more input sequences have been processed.

There are numerous applications for this system [1]. It could be used by autonomous robots. Such robots could be used as companions for people who need someone that can listen and reply sensibly. Natural language could also be used to obtain relevant facts or any kind of information previously stored. The system could also be used for vending machines to create a natural language interface. The probabilities created by the system could also be used to improve recognition rates of voice recognition systems [2]–[4]. Similarly, performance of natural language translation systems could also be improved [5]. The system could also be used for automatic text expansion [6] for people who are unable to talk.

II. Neural information processing

A standard model for neural information processing is the so called integrate and fire model [7]–[10]. The membrane potential of the neuron increases due to incoming signals. It increases until a certain threshold is reached. Once this happens, then the neuron "fires", i.e. the neuron sends an impulse along its axon. This signal reaches other neurons via synaptic junctions. Signals may also be exchanged via gap junctions [11], [12]. Other neurons also integrate their signals until they eventually fire. The strength with which neurons are connected to each other can be adjusted via Hebbian learning [13]. According to Hebbian learning, the synaptic strength increases if one neuron repeatedly causes another neuron to fire. Such large scale neural networks can be simulated using a computer [14].

III. Natural language processing

In Natural Language Processing, one tries to analyze and to understand natural language [15]–[18]. A number of researchers [19]–[21] have used deep neural networks to create captions for images. Kiros et al. [22] have used a neural network with long short-term memory [23] to encode language. Vinyals et al. [24] also used long short-term memory to generate natural sentences. Hermann et al. [25] have trained a deep neural network that can answer questions with respect to texts used during training. They also worked with long short-term memory. IBM Watson uses also Prolog in addition to techniques from natural language processing to answer questions from a very large knowledge domain [26]. Possible answers are rated using 354 features [27]. However, only a relatively small number of features (only 8) seem to be relevant to achieve an accuracy of 85.6% [27].

How can language or texts be stored in the weights of a neural network? How can texts be recalled from such a network? We will first look at known methods for information retrieval and natural language generation from computer science before we discuss our method for distributed storage and recall of texts. Manning [28] give an overview over methods for information retrieval. Essentially three different approaches exist for natural language generation: grammar-based, template-based or statistical approaches [29]–[31]. Grammar-based approaches [32], [33] need a grammar before language can

be generated or understood. But how can we learn such a grammar from listening to spoken language? Natural evolution may shape the neural hardware in order to support some sort of language. However, humans are not born with the ability to talk. Language needs to be acquired through learning, i.e. weights need to be tuned.

Template-based approaches work with pre-stored sentences or parts of sentences which contain placeholders at certain positions. These placeholders can be filled with one or several words in order to complete the sentence. Unfortunately, the range of possible sentences is limited. Only sentences similar to the pre-stored sentences can be generated. Statistical systems work with large text corpora. These corpora are statistically analysed. Probabilities are computed which determine how a partial sentence can be completed. The method which is presented here is also a statistical method. Other approaches use a semantic representation [34].

We will assume that language is solely learned from what is heard from the surrounding. The brain is a product of natural evolution. We also assume, that no information about any kind of grammar for a particular language is stored in the brain. Only counting is used to gather statistics on language usage. The mechanism supporting this data acquisition and storage could be considered a kind of universal "grammar" [35], [36]. However, it is not a grammar in the classical sense. It adapts to the current language. Using these assumptions, it should be possible to learn language from texts which are either heard or read. In other words, we will only use texts as input for our method.

IV. Using n-grams to store language

So called n-grams allow us to store and recall language in a distributed way. An n-gram is a sequence containing n distinct text components. The components could either be phones, syllables, letters or words. These n-grams are also used for clustering of web pages for web search [37] or for distinguishing different languages [38].

For the method described here, we will only consider words as building blocks. We parse each sentence into words. Then we compute the probability that one word follows the preceding $(n-1)$ words. This probability can be computed easily by taking a large text corpus, iterating over the individual words and counting how often each word follows the preceding sequence of $n-1$ words. Let w_i with $i \in \{1, ..., l\}$ be the individual words of a large text corpus with l words. Let $p(w_i|w_{i-(n-1)}, ..., w_{i-1})$ be the probability that word w_i follows the sequence $w_{i-(n-1)}$, ..., w_{i-1}. Given this probability, it is quite easy to complete a partial sentence. Similarly, we can take what has already been said in a conversation and create a new sentence taking the existing conversation as given. We only have to use the preceding $n-1$ words and compute the next word until all words of a sentence have been found. A simple method to find the next word is to only consider the k_c words with the highest probability. We simply perform a best first search. Then we continue with the k_c best alternatives and so on. We stop the search once an end of sentence marker has been found. Here, we will treat all punctuation marks as words. Hence, we stop the search as soon as we have reached a word marking the end of a sentence, i.e. a full stop, question mark or an exclamation mark. Using this method, we obtain a number of possible sentences which can all be used as a reply to what has been said before. Depending on the size of $n-1$, these sentences may be syntactically correct or not. A heuristic can be used to rate the sentences which have been generated. Finally, we pick the best sentence according to the heuristics in order to reply to what has been said so far.

This method can be implemented easily with a neural network (Fig. 1). Let us assume that we have one neuron per word in the word layer. This is the set of all possible words from our language. Whenever a sequence of $n-1$ words have been generated previously, then all neurons from the word layer, which correspond to a grammatically correct expansion of this partial sentence, should be activated. Their activation should be proportional to the probability with which the respective word follows the preceding $(n-1)$ words. On the n-gram level, we could use one neuron for each possible combination of $n-1$ words. Exactly one of these neurons will be activated when a sequence of $n-1$ words has been generated previously. The probability with which a certain word follows a given sequence of $n-1$ words could be stored in the weight from the neuron in the n-gram layer (representing the first $n-1$ words) to the neuron in the word layer (representing the next word). All of the weights between the n-gram layer and the word layer could be trained using Hebbian learning [13].

This approach could be used by the brain for natural language generation. The same method could also be used by a computer for natural language generation. Unfortunately, the method does not scale. Suppose that we only use 1-grams on the n-gram level. Then the word from the word layer would only depend on the last word that has been generated. If we were to use 2-grams, then the newly generated word would only depend on the last 2 words that have been generated, and so on. If we only use short n-grams inside the n-gram layer, then the newly generated words will have no relation to the beginning of the sentence. In other words, what has been said early on will have no influence on what is being said towards the end of the sentence. For this method to work, we will need very long n-grams. This would essentially mean that we store entire sentences within the n-gram layer. Let us assume that for every new word, we have 10 possible choices. That would lead to $O(10^n)$ neurons inside the n-gram layer. For $n = 11$ this would require all neurons of the brain to store possible sentence fragments.

Let us consider the average number of possible words that follows any n-gram for a sample corpus. This data is shown in Fig. 2. We can see that the average number of words that can follow any given n-gram goes down with n. Using neurons to represent n-grams would lead to storing entire sentence fragments for large n. According to this theory, we would have neurons for all possible sentences. Fig. 3(a) shows the number of n-grams for a given n. Fig. 3(b) shows the memory

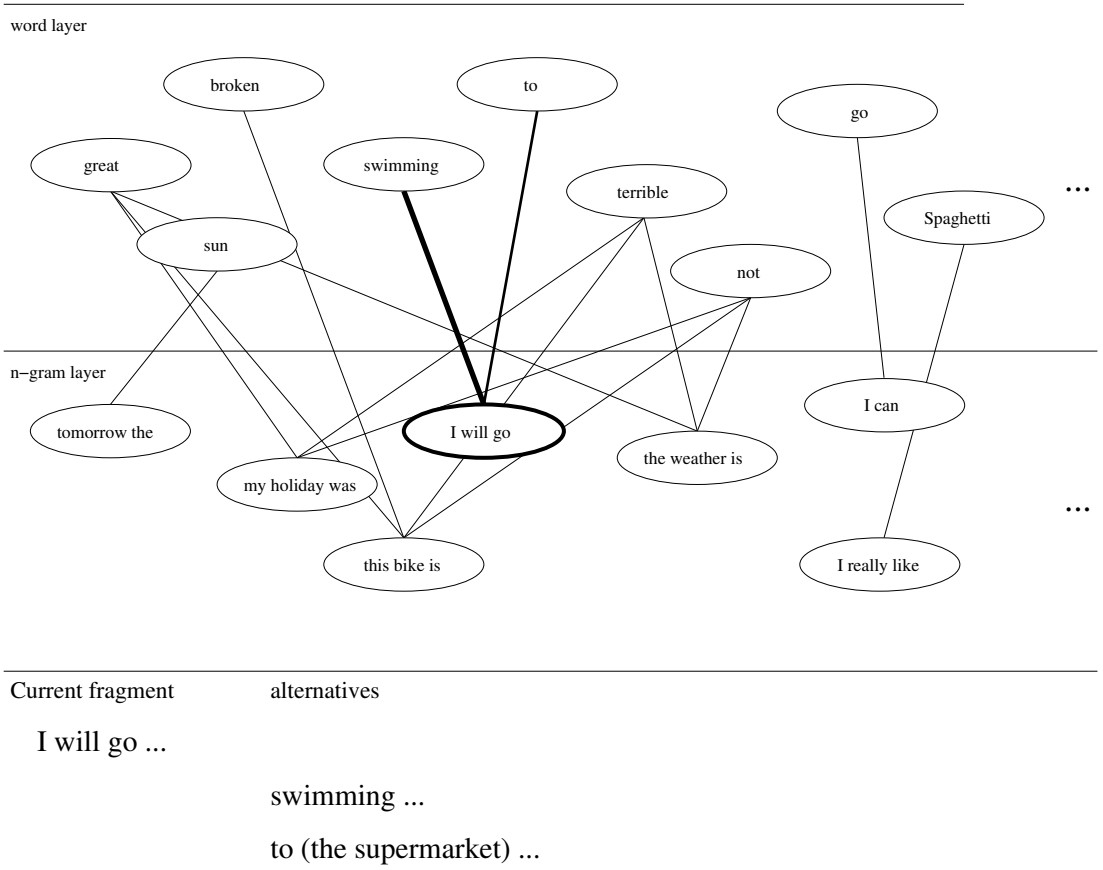


Fig. 1. Neural network for language generation. On the lowest layer (n-gram layer), neurons are activated whenever the words at the end of the sentence which has been generated so far, corresponds to the optimal stimulus of the neuron. On top of the n-gram layer is a layer of word neurons. The connection strengths correspond to the probability with which the respective word follows the n-gram level on the lower layer. In this example, the most likely alternative would be "swimming".

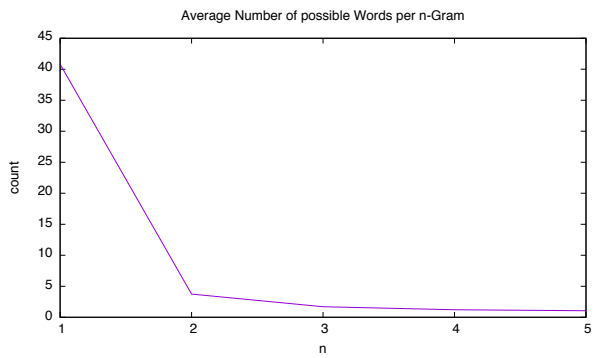
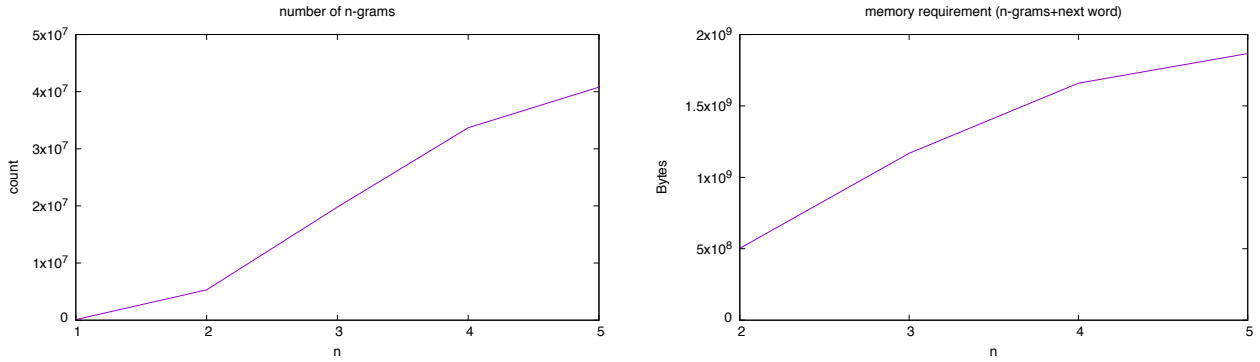


Fig. 2. Average number of words that can follow any n-gram for a sample corpus of 42.8 million words.

requirement if n-grams are used for sentence generation. It is assumed that n-grams plus words that can follow any given n-gram plus associated frequencies are stored. If we also consider the additional overhead due to the data structures used, then the memory requirement is so large that a 16 GB PC has to be used to store 2-grams, 3-grams, 4-grams and 5-grams plus the preceding and the next words from a corpus containing 42 million words.

V. Using delays to store language

Given the above, it is reasonable to assume that full sentences are not stored inside the brain. However, it is relatively easy to store $p(w_i|w_{i-1})$, i.e. the probability that word w_i follows word w_{i-1}. Similarly it would also be possible that the same probabilities but with a certain delay t with $t \in \{0, ..., t_{m-1}\}$, i.e. $p(w_i|w_{i-1-t})$ is stored in the brain (for a maximum possible delay m). This could be easily achieved by including additional neurons into the processing pipeline which cause a delayed processing of words. Fig. 4 shows this scenario where neurons are arranged in layers depending on the delay with which words are processed. Neurons at the bottom layer process what was generated early on. Neurons above the bottom layer process what was generated immediately after that and so on. Neurons at the top layer process the most recent word. The probabilities $p(w_i|w_{i-1-t})$ are stored in the weight from the neuron representing word w_{i-1-t} on the lower layer with delay t to the word w_i on the upper layer. For this approach, we would need $m + 1$ layers. Each layer would consist of n_{words} neurons, assuming our vocabulary consists of n_{words}. Total number of neurons is now $O(mn_{\text{words}})$



(a) (b)

Fig. 3. (a) Number of n-grams for a given n. (b) Memory requirement for sentence generation using n-grams. It is assumed that n-grams plus words that can follow a given n-gram plus associated frequencies are stored.

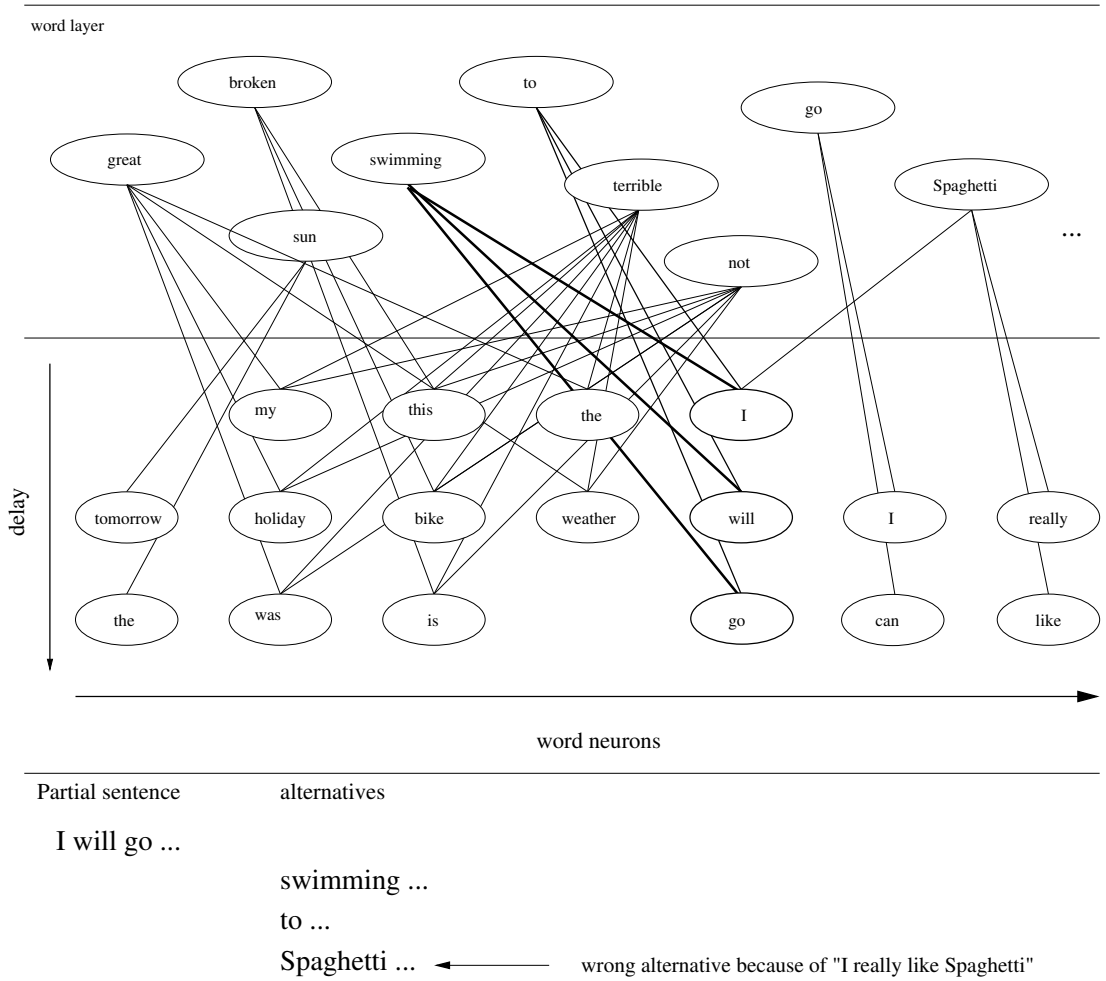


Fig. 4. Neural network for natural language generation. Neurons on the lower word layer will be activated whenever the corresponding word was either heard or read with a given delay. The delay increases as we move downward. Taken together, one word from each layer represents a sequence of words. On the upper layer we have a neuron for every word from our language. The weights from the lower to the upper layer correspond to the probability with which the word follows the given word/delay inside the lower layer. Here, the most likely alternative would be “swimming”.

which is a considerable reduction in the number of neurons required compared to the n-gram approach. The delay network approach could be used by the brain in terms of number of neurons required. Unfortunately, the number of words which could follow word w after a delay t increases with the delay t. This is illustrated in Fig. 5 for a sample corpus with 42.8 million words. Fig. 5(a) shows the number of words that follow a given word averaged over all words of the vocabulary for delays 1 to 10. Fig. 5(b) shows the number of words that follow a given word for delays 1 to 10. It is clear that words next to a given word are more correlated or constrained while words further away are effectively uncorrelated.

The larger the delay i, the less certain we are, which word w_i should be placed at a distance t from word w_{i-1-t}. An advantage of this approach is that it scales. The number of neurons increases linearly with the number of delay layers. Unfortunately, the information stored in $p(w_i|w_{i-1-t})$ is not sufficient to produce grammatically correct sentences. We have

$$p(w_i|w_{i-m},...,w_{i-1}) \neq \Pi_{j=0}^{m-1} p(w_i|w_{i-1-j}). \quad (1)$$

In other words, we do not have a Markov Model where the probability of obtaining the next state depends only on the current state. Because of this, the approach also does not seem to be used by the brain. If we only consider individual letters, then a Markov model of length 6 seems to be sufficient in order to create a sequence of correctly written words [38]. Unfortunately, this sequence of words is not grammatically correct.

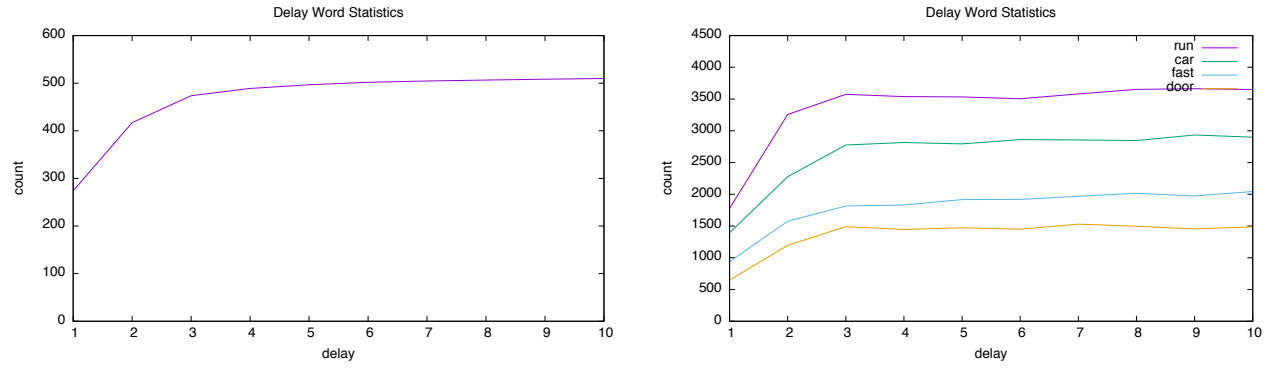
VI. Distributed storage of sentences

Some other method of storing sentences inside the weights of a neural network is required. The following observation allows us to successfully store and recall sentences using a distributed neural network: The relative frequency of words within a sentence is approximately constant over a relatively long time while language is being processed. Fig. 6 shows the frequency of words for a sample corpus of 42.8 million words. The most common word is the full stop with a frequency of 4.591.872. The second most common word is "the" with a frequency of 3.046.404. Because we treat full stops and commas as words, "," is the third most common word. The words "to", "a", "and", "I", and "of" are ranked with descending order. Starting from position 300, we find the words "thank, "guess", "games" and "top". From position 2.000, we find the words "choices", "mainly", and "familiar". From position 10.000, we find the words "festivals", "collectively", "mushrooms" and "inexpensive". From this, it is quite clear that certain words occur more frequently in one's vocabulary than other words. Of course, the article "the" ist the most frequent word apart from the full stop. Other words occur less frequent. This is the well known Zipf's law [28]. According to this law, the frequency of any word is inversely proportional to its rank as shown in Fig. 6.

If we process natural language with a computer, we simply need to count how often each word occurs in daily usage. In the brain, we could simply take one neuron for each word in our vocabulary. Using Hebbian learning [13], we could train the weights to encode word frequencies. Even if these frequencies change during further processing/learning, the relative order of the frequencies of the words within a single sentence remains constant.

Let us now consider the sample sentence "I hope that tomorrow the sun will shine." The frequencies of the individual words are: "I (1.512.476) hope (25.619) that (1.037.840) tomorrow (4.103) the (3.046.404) sun (4.968) will (193.675) shine (730) . (4.591.872)". It is not very likely that the frequency of the word "sun" will become larger than the word "the" during additional learning/language processing. Also, the words with the lowest frequency tell us what the sentence is about. In this example the words with the lowest frequency are "hope tomorrow sun shine". In other words, low frequency words are especially meaningful. Interestingly, the same words would be extracted if the sentence were German. Words which occur with high frequency are called stop words [28]. Of course, stop words are necessary to construct a grammatically correct sentence. However, they are not required to understand the general meaning of a sentence.

We will now have a look, at how we can make use of this observation for storage and recall of texts or sentences. Whenever we process a word that has never been seen before, we increase a counter. We will use indices to represent words. End of sentence markers like full stop, exclamation mark and question mark are assigned indices starting from 0. The index after the end of sentence markers will be assigned the comma. In other words, punctuation marks will have the lowest indices. Indices after the comma will be used for words. Whenever we encounter a new word, we will use a new index for that word. For each word, we count how often it has been processed, i.e. has been heard or read. Thus, we obtain word frequencies $f(w)$ for words w where w is the index of the word. The method will stop processing after a certain number of sentences have been processed. All words which occur less than a certain number of times could be removed. Words that occur infrequent are probably not very relevant to our vocabulary. It may also have been a typo. We do not want to include typos into our vocabulary. During this update, the method is not able to process input. This is similar to the requirement for sleep in humans. However, we do not want to suggest that the processing occurring during sleep is the same update that we perform here. Nevertheless, it is quite interesting that this method also needs to "sleep" periodically to perform an update. For the experiments below, we perform this update whenever our vocabulary exceeds 130000 words. We sort all words according to their frequencies and simply remove low frequency words. After sorting, the most frequent word will have the lowest word index. Low frequency words will have high indices. If we sort the words according to their frequencies, then we can use the indices to find out whether one word occurs more frequent than another word by simply comparing their indices. If we do not sort the words, then we have to look up the frequency for a word in order to determine whether one word occurs more frequent than



(a) (b)

Fig. 5. (a) Number of words that follow a given word averaged over all words of the vocabulary. (b) Number of words that follow four sample words ("run", "door", "fast" and "car").

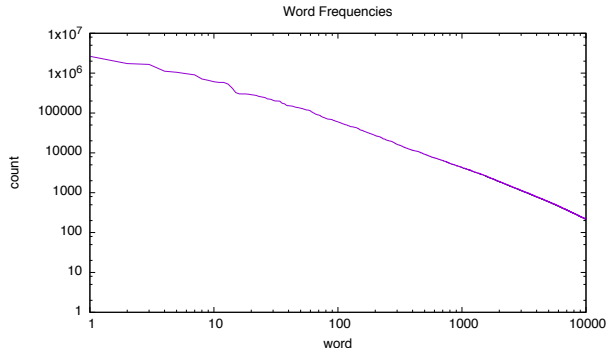


Fig. 6. Frequency of words in a corpus of 42.8 million words. Word frequency is inversely proportional to its rank. This is the well known Zipf's law.

another word. In order to keep memory requirements down, we limit our vocabulary to a fixed number $n_{\max}$ of words. This is the approach that we have used for our experiments. Only the 100.000 words which occur most frequent are kept in our vocabulary.

We can compute probabilities $p(w_i|w_{i-1})$, $p(w_i|w_{i-2}, w_{i-1})$ and $p(w_i|w_{i-3}, w_{i-2}, w_{i-1})$ by counting how often the word w_i follows the word w_{i-1}, the sequence of words w_{i-2}, w_{i-1}, or the sequence of words $w_{i-3}, w_{i-2}, w_{i-1}$. Using $p(w|".")$, we can find any word which can start a sentence. Whenever we only have a single word w_1, then we can use $p(w|w_1)$ to find words which could follow word w_1. If we have two words w_1, w_2 then we can use $p(w|w_1, w_2)$ to find possible words to extend the sequence w_1, w_2. Similarly, if we have a sequence of three words w_1, w_2, w_3, then we can use $p(w|w_1, w_2, w_3)$ to extend the sequence. Using this approach, all generated sequences which consists of up to 4 words are guaranteed to be grammatically correct. Unfortunately, sequences longer than 4 words may not be grammatically correct.

Using this approach, how can we create a relation between what is being said towards the end of the sentence with the beginning of the sentence? We simply take advantage of the fact that low frequency words are especially meaningful. We can use them and omit stop words, i.e. words which occur with a high frequency, to extend the range of probabilities $p(w|w_1, ..., w_i)$. We will call these probabilities $\hat{p}$. In order to store sentences using the probabilities $\hat{p}$, we iterate over all words of a sentence. We keep track of the words with the smallest frequencies. Let w_{m1} be the word with the smallest frequency among the words seen in the sentence so far. Let w_{m2} be the word with the second smallest frequency among the words seen in the sentence so far. We will use the words w_{m2} and w_{m1} in addition to the last word that has been processed in order to find alternative words to append to this sentence. We will store probabilities $\hat{p}(w|w_{m2}, w_{m1}, w_i)$. Fig. 7 shows how these probabilities are updated for a sample sentence "I hope that tomorrow the sun will shine.". In practice, it is sufficient, so simply count how often a certain word follows a given tuple.

Let $w_1, ..., w_i$ be a sequence of words. We can use $p(w|w_{i-2}, w_{i-1}, w_i)$ to find alternative words to expand the existing sentence. Given the sentence fragment $w_1, ..., w_i$, we can extract the two words w_{m1} and w_{m2} from this fragment, which occur least frequent in our vocabulary. Therefore, $\hat{p}(w|w_{m2}, w_{m1}, w_i)$ as well as $p(w|w_{i-2}, w_{i-1}, w_i)$ present us with a list of possible words wich can be used to extend the sentence. We will only allow words to expand the given sentence which have a non-zero probability due to $p(w|w_{i-2}, w_{i-1}, w_i)$ and $\hat{p}(w|w_{m2}, w_{m1}, w_i)$. We will use a conservative approach. We use the minimum probability of both $p(w|w_{i-2}, w_{i-1}, w_i)$ and $\hat{p}(w|w_{m2}, w_{m1}, w_i)$ in order to find a word to expand the given sentence, i.e. we compute

$$p(w|w_1, ..., w_i) \approx \min\{p(w|w_{i-2}, w_{i-1}, w_i), \hat{p}(w|w_{m2}, w_{m1}, w_i)\}. \quad (2)$$

Instead of using the minimum, we could also work with the sum of these probabilities (maintaining zero probability) or we could compute the product of these probabilities. It would also be possible to use log probabilities. However, for the experiments below, we have used the minimum-operation.

This approach could also be implemented by a neural network. In order to find the word with the smallest frequency

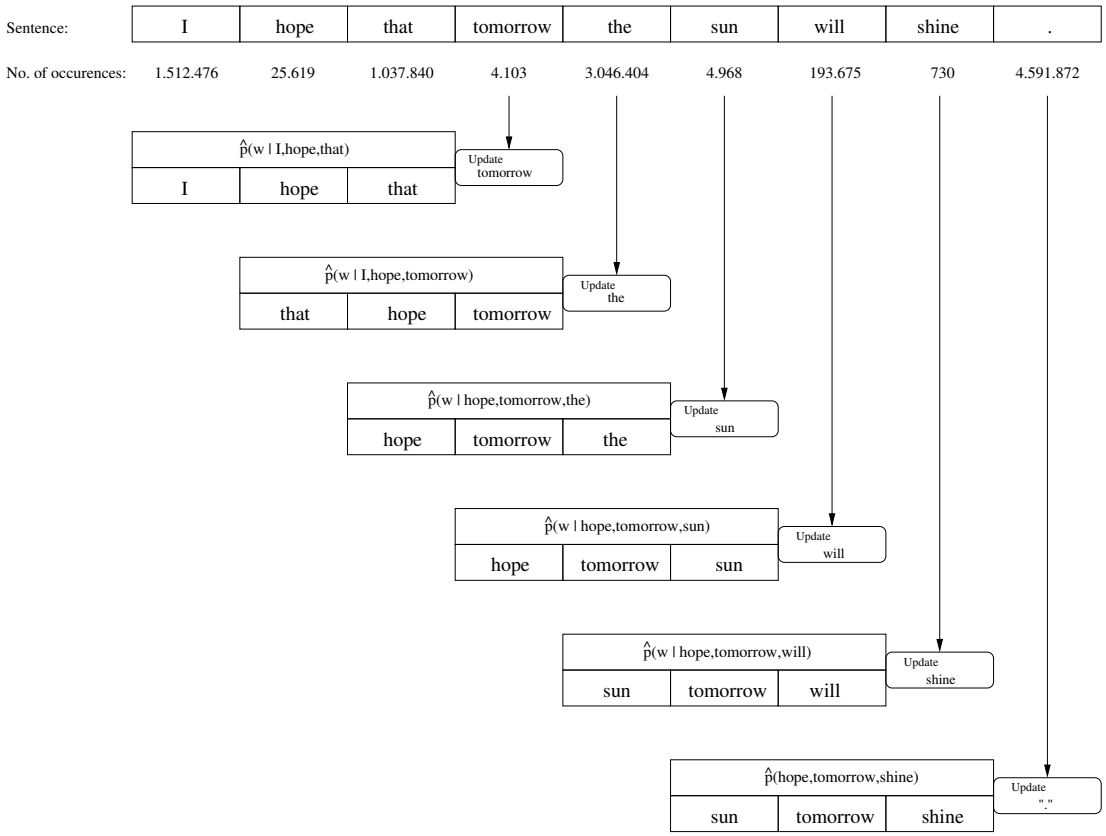


Fig. 7. Sample updates of probabilities $\hat{p}(w|w_{m2}, w_{m1}, w_i)$ for a sample sentence.

among the words that have been processed from the sentence so far, we iterate over all words sequentially. We would need memory to store the frequency of the current word. It would get updated whenever the frequency of the current word is less than what is already stored in this memory. Whenever a new sentence starts, this memory would be initialized to the frequency of the first word of the sentence. All of this could be achieved with a set of neurons, one neuron for each word of our vocabulary. The frequency with which these words occur in our daily language could be stored in the weights to another neuron. These weights could be trained through Hebbian learning [13]. This neuron would be activated by the current word which is processed. The activation of this neuron could be transferred to another neuron but only if the activation is less than the current activation of that neuron. This could be achieved by first inverting the signal (now larger represents higher frequency) and then taking the maximum of the current activation stored by this neuron and the new value. This maximum operation could be performed using a softmax function. Alternatively, Long Short-Term Memory [23] could be used. Long Short-Term Memory would be updated whenever the signal is larger. It would be reset to zero, when a new sentence starts. An alternative to inverting the signal would be to have neurons arranged depending on their usage during language. Frequently used neurons would be located in one part of the brain, while neurons coding for words heard less often would be stored in another part of the brain. Neurons coding for words less often would have a higher activation because they are located in a different area. This variant is similar to using word indices where low index words represent words which are often heard or processed, while high index words represent words which are less frequently heard or processed.

VII. Recall of sentences

Using the above, it is quite easy to construct a system which is capable of natural language processing. Suppose that we have a fully trained system and that we pose a question or assertion to this system. Let $w_1, ..., w_n$ be this sentence. From this sentence, we can extract the k words which have the lowest frequency within the words of this sentence. These sentences represent the gist of what has been said so far. We can use these words to recall sentences which have been stored in the system in a distributed way. This can be done by taking all k words as seeds or starting words for sentence generation. Whenever a sentence has to be extended, we search for possible alternatives. For each new words, we allow maybe three to four alternatives. Allowing for four alternatives, we would obtain $4^3 = 64$ possible sentences after finding three more words. However, with increasing length of the sentence, the number of alternatives gets reduced. It is possible that only a single alternative is grammatically correct given the

probabilities stored by the system. Starting from a seed word, we continue until an end of sentence marker has been reached, i.e. a full stop, exclamation mark or question mark.

This leads to a number of sentence fragments. Each fragment starts with one of the seed words. In order to obtain a full sentence, we need to expand it towards the beginning of the sentence. We need all of the above probabilities in opposite reading order. Let $p_r(w|w_1,...,w_n)$ be the probability that word w follows the words $w_1,...,w_n$ in normal reading order. In the same way that we have computed $p_r(w|w_1,...,w_n)$, we can compute $p_l(w|w_1,...,w_n)$, the probability that word w precedes the sequence $w_1,...,w_n$. Given p_r and p_l, we can now extend a sentence in any direction. Any fragment of a sentence can be completed until an end of sentence marker has been reached.

Given any kind of input, which we call the original input, we can extract k seed words. Starting with these seed words, we can generate a number of possible sentences using the probabilities p_l and p_r. Then, we can use heuristics to decide which sentence would be suitable as a reply to the original input. For our experiments we have used the following 5 different heuristics to provide a quantitative rating of all sentences. 1) sum of word-sequence-probabilities, i.e. the probability that one word follows or precedes a given word of the sentence. 2) sum of n-gram probabilities over all n-grams of the sentence. 3) information content of the sentence, i.e. sum of word frequencies of all words of the sentence. 4) similarity of the sentence to the previous sentence. One point per word of the original sentence which also occurs in the generated sentence. 5) Fit of the generated sentence to the topic of the original sentence.

For each heuristic we compute a quantitative score s_i with $i \in \{1,...,5\}$. We now fully describe how these scores are computed for a complete sentence $w_1,...,w_n$ as a reply to the original input $w'_1,...,w'_{l_1}$.

Sum of word-sequence-probabilities s_1:

$$s_1 = \sum_{i=2}^{n} (p_r(w_i|w_{i-1}) + p_l(w_{i-1}|w_i)) \tag{3}$$

Sum of n-gram probabilities s_2:

$$s_2 = \sum_{i=4}^{n} (p_r(w_i|w_{i-3},w_{i-2},w_{i-1}) + p_l(w_{i-3}|w_{i-2},w_{i-1},w_i)) \tag{4}$$

Information content of the sentence s_3. Word indices are sorted in order of decreasing frequency. Hence, we have $w_i > w_{i-1}$ if $p(w_i) < p(w_{i-1})$.

$$s_3 = \sum_{i=1}^{n} w_i \tag{5}$$

Similarity of the sentence to the previous sentence. s_4: Let $w''_1,...,w''_{l_2}$ be the unique words of the sentence $w'_1,...,w'_{l_1}$, then

$$s_4 = \sum_{i=1}^{n} \sum_{j=1}^{l_2} \delta(w_i, w''_j) \tag{6}$$

where δ is the Kronecker delta.

Fit of the generated sentence to the topic of the original sentence s_5. Let $w^s_1,...,w^s_k$ be the k words of the original sentence $w'_1,...,w'_{l_1}$ with the least frequency. The word w^s_1 is the word from the original sentence with the smallest frequency in our vocabulary.

$$s_5 = \sum_{i=1}^{n} \sum_{j=1}^{k} (k-j+1)\delta(w_i, w^s_j) \tag{7}$$

We normalize each score $s_i(a)$ that has been computed for sentence $a = w_1,...,w_n$, by dividing each score by the maximum of all scores over all generated sentences. Then we sum up all normalized scores, obtaining $s(a)$.

$$s(a) = \sum_{i=1,...,5} \frac{s_i(a)}{\max_b s_i(b)} \tag{8}$$

Instead of normalizing by the maximum score, it would also be possible to normalize by dividing scores by the sum of scores. Finally, we chose the sentence a with highest score $s(a)$ among the generated sentences as a reply to the original sentence $w'_1,...,w'_{l_1}$. It would also be possible to use only a subset of the five heuristics presented here. A weighted sum could also be used to compute the overall score $s(a)$. This would allow us to give more weight to certain more relevant heuristics in relation to others that are less relevant.

The complete model is shown in Fig. 8. The input sequence is converted into individual words. All words are analyzed based on their frequency. The most relevant words (least frequent words) are memorized. For each of these words, a sentence is generated based on statistical data which has been gathered so far. Among these sentences, one sentence is finally selected.

VIII. Materials and methods

In order to test the above method for storing and recalling sentences, a corpus of 59 million words has been used. The corpus has been extracted from the site Reddit.com, which is a popular discussion forum. All comments from the subreddits art, askreddit, askscience, books, diy, documentaries, europe, explainlikeimfive, food, funny, futurology, gadgets, gaming, getmotivated, history, internetisbeautiful, jokes, lifeprotips, listentothis, mildlyinteresting, movies, music, news, personalfinance, philosophy, science, sports, television, todayilearned, upliftingnews, videos, worldnews, which were available through the web page on 5 March 2017 have been included in the corpus (16.2 million words). Additionally, all comments from the "Complete Public Reddit Comments Corpus (SQLite)"[1] posted on 1 July 2014 have also been used for learning (42.8 million words). Together, a corpus of 59 million words have been used for training. The method was implemented using unordered maps in C++ to store the frequency with which a given word follows a certain tuple of words.

[1] https://archive.org/details/2015_reddit_comments_corpus_sqlite

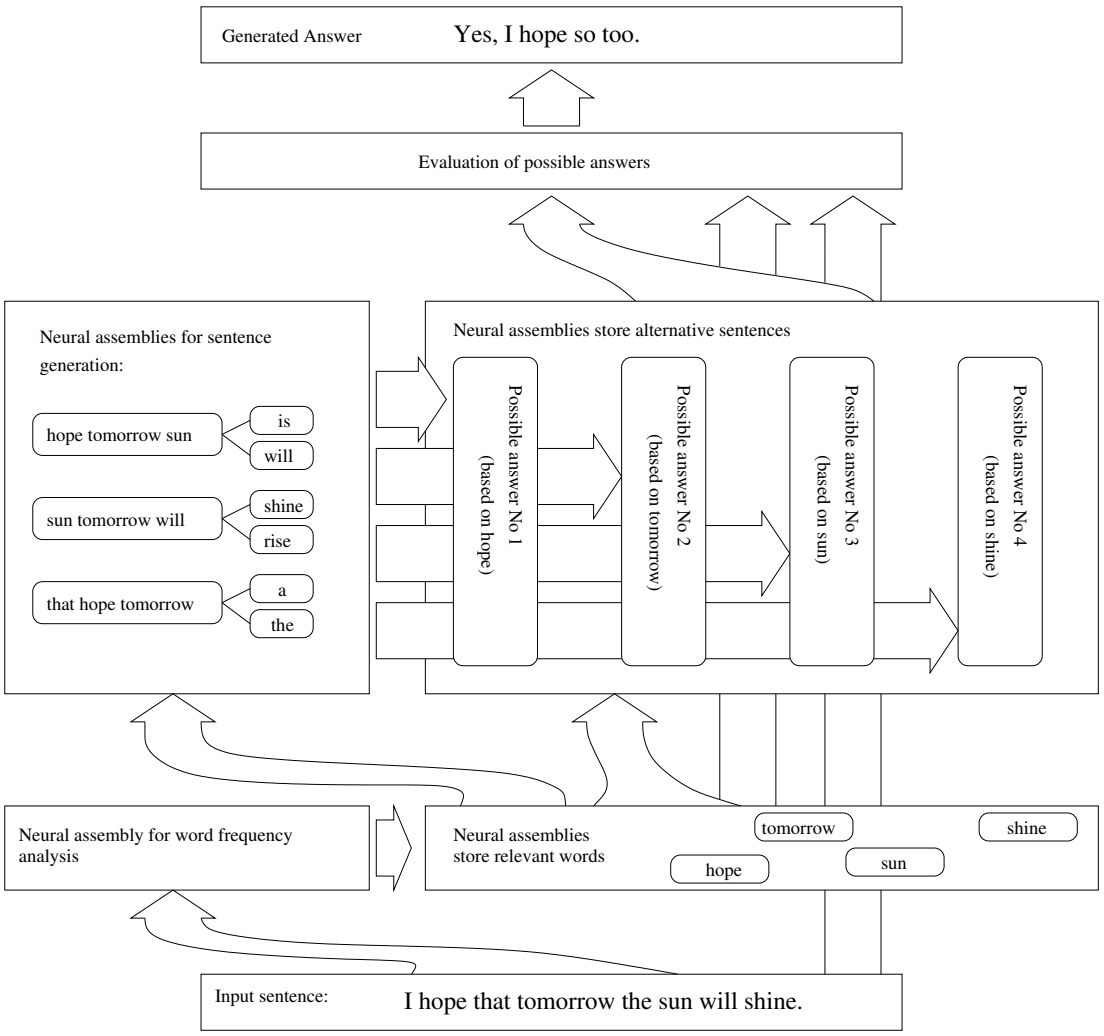


Fig. 8. Neural model for language generation. As a first step, the input is separated into words. Each word is analyzed depending on its frequency during natural language usage. Low frequency words are memorized and sentences are generated based on these words. Finally, a single sentence is chosen which is then output.

A number of sample questions and assertions will be addressed to this system. For each question, we show the best answers sorted with decreasing score $s(a)$. The punctuation mark of the preceding sentence is also included in the computation. Additional answers which differ only in the punctuation mark of the preceding sentence have been omitted. We take our questions and assertions from a quite general subject area. The answers, that have been given by the system, are stored somewhere almost verbatim in the corpus. All words are written in small letters. No distinction was made between lower case and capital letters in order to save memory space. The parameter k was set to 3. In other words, only 3 words are used to describe the gist of the original sentence. When expanding sentences, we only consider the two alternatives which are most likely. Additionally, we also allow up to two more alternatives if these alternatives are among the k most relevant words from the original input sentence. Most of the answers created with this system are reasonable replies to the original sentence.

- Statement: i like the beatles .
  - if you like rock music , you'll like the beatles .
  - i like the way it is . (Corpus contains "and I don't like the way it is")
  - i like the way it looks . (Corpus contains "I just like the way it looks.")
  - i like the way you think .
- Statement: city life is not better than country life .
  - this is exactly what's wrong with this city country right now . (Corpus contains "This is EXACTLY what's wrong with this city/country right now.")
  - you're not from chicago and you don't know jack shit about city life .
  - what specifically draws you to city life ?
  - i'm english and know the area the book was set in the book was set in the south west of england i thought it was a pretty decent portrayal of country life . (Corpus

contains “I’m English and know the area the book was set in relatively well (I live in the South West of England) - I thought it was a pretty decent portrayal of country life.”)

- Question: is the country a better place to live ?
  - a place to live is a good start .
  - i want to live . (Corpus contains “My only goal right now is to manage my depression well enough to be able to function in the life I want to live.”)
  - i live near the mountains .
  - i live near the border . (Corpus contains “I used to live near the border and miss it :p)”)
- Question: what is better , democracy or a dictatorship ?
  - i think the alternative is benevolent dictatorship . (Corpus contains “I think the alternative is r/Bitcoin.”)
  - with more recent reforms giving better investment and autonomy along with breaking the region up into better administered provinces a lot of the tensions have been diminished from earlier years when indonesia was a military dictatorship .
  - what tb wants is a benevolent dictatorship . (Corpus contains “What TB wants is a benevolent dictatorship, and everyone knows that just in practice does not work.”)
  - this is the mark of a dictatorship .
- Statement: rock music is great .
  - rock , folk , blues , jazz , and pretty much every genre of music ever . (Corpus contains “rock, folk, blues, jazz, electronic, hip-hop, world, punk, metal, soul.”)
  - this clipping makes the distortion sound we’re all familiar with in rock music .
  - his harmonies are really fucking intelligent on a level you don’t usually hear in rock music .
  - the music is great . (Fragment “the music is great” contained multiple times in corpus)
- Question: why are so many people unemployed ?
  - &gt , why do we have a ton of people who are overworked doing the jobs of 2 3 people and then also have millions of unemployed who need work ? (Corpus contains “Why do we have a ton of people who are overworked doing the jobs of 2-3 people and then also have millions of unemployed who need work?”)
  - you have a very unique play style to do well with , and that’s why many people don’t like them .
  - also , and that’s why many of us are not . (Not in corpus)
  - obviously , there are diminishing returns which is why it isn’t 1 person does 80 hours and 3 people are unemployed .
- Statement: university education is expensive .
  - i am in the second semester of my freshman year at a fairly expensive university .
  - i’m guessing it doesn’t require a university education .
  - a university is not a criminal court , the student doesn’t risk getting jailed at the end of the school proceeding , and so the school isn’t required to adopt a criminal standard of evidence .
  - the full name of the university . (Corpus contains “ The only time I ever use it for us, however, is when I am using the full name of the university.”)
- Statement: students do not learn as much as they should .
  - why should students then study literature and not media more broadly ?
  - i should learn to play the game .
  - students should make wise borrowing decisions .
  - the students critically developing how to obtain the answers they seek succeed later in life .

IX. Conclusions

The above system is able to store and recall sentences. The system processes any kind of input sentence, extracts the relevant words and generates a seemingly meaningful answer sentence. Internally, several alternative answer sentences are generated and subsequently scored based on the original input sentence. Only statistical data gathered from previously processed input sentences is used to generate answer sentences and to score them. The system essentially tokenizes the input sentence into words and generates n-tuples with $n \leq 3$ from these words. Some of these tuples contain sequentially ordered words as they appeared in the input sentence while the remaining tuples may be considered a summary of the sentence processed so far. For each tuple, it is simply counted how often a particular word occurs as an alternative for the next word to be read or generated. Using only information from these 3-tuples, it is possible to generate quite long answer sentences.

The system could be used to create autonomous robots that can interact with humans in a natural way. Humans would be able to address such robots using a natural language interface and would obtain a reply in a similar way that a human would respond (more or less relevant to the original input). Additionally, this system could also be used to improve voice recognition systems or natural language translation systems because the system generates plausible alternative words for sentence fragments which have been obtained so far. Given such alternative words, it is possible to reduce the number of possible meanings and hence to simplify the task of voice recognition or automatic translation. Finally, people who are unable to talk, could use it for automatic text expansion.

Conflict of interest statement

Author contributions: The author invented the method, implemented the software and ran the experiments. Research funding: Ernst Moritz Universität Greifswald. Employment or leadership: Marc Ebner is Professor of Computer Science at

the Ernst Moritz Arndt Universität Greifswald. Honorarium: The funding organization played no role in the study design; in the collection, analysis, and interpretation of data; in the writing of the report; or in the decision to submit the report for publication.

REFERENCES

[1] M. Ebner, "Verfahren und Vorrichtung zum maschinellen Verarbeiten von Texten," *Deutsche Patentanmeldung, 42 Seiten, 21. Dezember, DE P16025DE02*, 2016.

[2] G. H. Abrego and X. Menendez-Pidal, "Supervised automatic text generation based on word classes for language modeling," *United States Patent US 7,035,789*, Apr. 2006.

[3] B. D. Metz, "Automatic grammar tuning using statistical language model generation," *United States Patent Application US 2008/0052076 A1*, Feb. 2008.

[4] W. W. Chang, G. D. Wilensky, and L. A. Dontcheva, "Natural language vocabulary generation and usage," *United States Patent Application US 2014/0081626 A1*, Mar. 2014.

[5] O. Christ, "Dynamic generation of auto-suggest dictionary for natural language translation," *United States Patent Application US 2011/0184719 A1*, Jul. 2011.

[6] V. Fux and M. G. Elizarov, "Handheld electronic device and method for disambiguation of compound text input and that employs n-gram data to limit generation of low-probability compound language solutions," *United States Patent US 8,265,926*, Sep. 2012.

[7] H. R. Wilson and J. D. Cowan, "Excitatory and inhibitory interactions in localized populations of model neurons," *Biophysical Journal*, vol. 12, pp. 1–24, 1972.

[8] W. M. Kistler, W. Gerstner, and J. L. van Hemmen, "Reduction of the Hodgkin-Huxley equations to a single-variable threshold model," *Neural Computation*, vol. 9, pp. 1015–1045, 1997.

[9] H. R. Wilson, "Simplified dynamics of human and mammalian neocortical neurons," *J. theor. Biol.*, vol. 200, pp. 375–388, 1999.

[10] C. Christodoulou, G. Bugmann, and T. G. Clarkson, "A spiking neuron model: Applications and learning," *Neural Networks*, vol. 15, no. 7, pp. 891–908, Sep. 2002.

[11] M. Ebner and S. Hameroff, "Lateral information processing by spiking neurons – a theoretical model of the neural correlate of consciousness," *Computational Intelligence and Neuroscience*, 2011.

[12] ——, "Modeling figure/ground separation with spiking neurons," in *Simulation in Medicine – Preclinical and Clinical Applications*, I. Roterman-Konieczna, Ed. Berlin: de Gruyter, 2015, pp. 77–96.

[13] D. O. Hebb, "The organization of behavior, chapter 4, new york, wiley, 1949," in *Neurocomputing: Foundations of Research*, J. A. Anderson and E. Rosenfeld, Eds. Cambridge, Massachusetts: The MIT Press, 1988.

[14] E. M. Izhikevich and G. M. Edelman, "Large-scale model of mammalian thalamocortical systems," *Proceedings of the National Academy of Sciences USA*, vol. 105, no. 9, pp. 3593–3598, 2008.

[15] A. B. Carus, M. Wiesner, and K. Boone, "Method and apparatus for morphological analysis and generation of natural language text," *United States Patent US 5,794,177*, Aug. 1998.

[16] C. P. Rehberg, "Automatic pattern generation in natural language processing," *United States Patent US 8,180,629*, May 2012.

[17] J. E. Bostick, J. M. Ganci, Jr, J. P. Kaemmerer, and C. M. Trim, "Ontology driven dictionary generation and ambiguity resolution for natural language processing," *United States Patent US 9,372,924*, Jun. 2016.

[18] C. D. Manning and H. Schütze, *Foundations of Statistical Natural Language Processing*. Cambridge, Massachusetts: The MIT Press, 1999.

[19] S. Chengjian, S. Zhu, and Z. Shi, "Image annotation via deep neural network," in *International Conference on Machine Vision Applications, Tokyo, Japan*, May 2015, pp. 518–521.

[20] H. Fang, S. Gupta, F. Iandola, R. Srivastava, L. Deng, P. Dollar, J. Gao, X. He, M. Mitchell, J. Platt, L. Zitnick, and G. Zweig, "From captions to visual concepts and back," in *Proceedings of Computer Vision and Pattern Recognition*. IEEE, Jun. 2015.

[21] R. Kiros, R. Salakhutdinov, and R. Zemel, "Multimodal neural language models," in *Proceedings of the 31st International Conference on Machine Learning, Beijing, China*, 2014.

[22] ——, "Unifying visual-semantic embeddings with multimodal neural language models," in *TACL*, 2015.

[23] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[24] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," in *Proceedings of Computer Vision and Pattern Recognition*. IEEE, 2015, pp. 3156–3164.

[25] K. M. Hermann, T. Kociský, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom, "Teaching machines to read and comprehend," in *Advances in Neural Information Processing Systems (NIPS)*, 2015. [Online]. Available: http://arxiv.org/abs/1506.03340

[26] A. Lally and P. Fodor, "Natural language processing with prolog in the ibm watson system," *Association for Logic Programming*, 2011.

[27] U. Bhowan and D. J. McCloskey, "Genetic programming for feature selection and question-answer ranking in ibm watson," in *Proceedings of the 18th European Conference on Genetic Programming, Denmark, April 8–10*, P. Machado, M. L. Heywood, J. McDermott, M. Castelli, P. García-Sánchez, P. Burelli, S. Risi, and K. Sim, Eds. Berlin: Springer, 2015, pp. 153–166.

[28] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. New York, NY: Cambridge University Press, 2008.

[29] F. Schilder, "Systems and methods for natural language generation," *United States Patent Application US 2014/0149107 A1*, May 2014.

[30] B. Howald, R. Kondadadi, and F. Schilder, "Systems and methods for natural language generation," *United States Patent US 9,424,254*, Aug. 2016.

[31] A. Kaeser, E. Vignon, and L. Stoecklé, "Systems and methods for natural language generation," *United States Patent US 9,411,804*, Aug. 2016.

[32] S. Bangalore and O. C. Rambow, "System and method for natural language generation," *United States Patent US 7,562,005*, Jul. 2009.

[33] A. Ratnaparkhi, "Trainable dynamic phrase reordering for natural language generation in conversational systems," *United States Patent Application US 2002/0116173 A1*, Aug. 2002.

[34] S. Pan and C.-K. Shaw, "Method, program, and apparatus for natural language generation," *United States Patent US 7,496,621*, Feb. 2009.

[35] N. Chomsky, "Three models for the description of language," *IRE Transactions on Information Theory*, vol. 2, no. 3, pp. 113–124, Oct. 1956.

[36] M. H. Christiansen and N. Chater, "Language as shaped by the brain," *Behavioral and Brain Sciences*, vol. 31, no. 3, pp. 489–558, 2008.

[37] A. Z. Broder, S. C. Glassman, M. S. Manasse, and G. Zweig, "Syntactic clustering of the web," *SRC Technical Note*, Jul. 1997.

[38] T. Dunning, "Statistical identification of language," *Technical Report MCCS 94-273*, Mar. 1994.