Vector-to-Sequence Models for Sentence Analogies

Liyan Wang
Graduate School of Information, Production and Systems
Waseda University
Kitakyushu, Japan
wangliyan0905@toki.waseda.jp

Yves Lepage
Graduate School of Information, Production and Systems
Waseda University
Kitakyushu, Japan
yves.lepage@waseda.jp

***Abstract*—We solve sentence analogies by generating the solution rather than identifying the best candidate from a given set of candidates, as usually done. We design a decoder to transform sentence embedding vectors back into sequences of words. To generate the vector representations of answer sentences, we build a linear regression network which learns the mapping between the distribution of known and expected vectors. We subsequently leverage this pre-trained decoder to decode sentences from regressed vectors. The results of experiments conducted on a set of semantico-formal sentence analogies show that our proposed solution performs better than a state-of-the-art baseline vector offset method which solves analogies using embeddings.**

***Index Terms*—sentence analogies, decoder, sentence embeddings.**

I. Introduction

Analogy has played an important role in symbolic artificial intelligence because of its potential to capture structural relations. It was also felt as a possible way of imitating human cognition [1], [2], [3]. Specifically, given a pair of entities (A, B) in a source domain and an entity C in a target domain, the answer D is inferred by mapping relational features or structural correspondences across the two domains, so as to satisfy the analogical relations as much as possible. This gives rise to an analogy, noted $A : B :: C : D,$ like the one in $\textit{reservoir} : \textit{water} :: \textit{battery} : \textit{electricity}$ from the domain of standard physics to that of electromagnetism. Analogical reasoning has proved to be an effective problem-solving method in many tasks, such as image generation [4] or preference completion [5].

Due to its ability to reflect and capture linguistic features, analogical reasoning has been used to complete some Natural Language Processing (NLP) tasks, like machine translation [6], [7], [8] or question answering [9] in recent years. In NLP, the terms of an analogy are typically words, sentences, or any other textual units. For instance,

$$\textit{male} : \textit{king} :: \textit{female} : x \quad \Rightarrow \quad x = \textit{queen}$$

(to tediously quote the over-repeated example from [10]) or

$$\begin{array}{l}\textit{A coffee,}\\ \textit{please!}\end{array} : \begin{array}{l}\textit{A strong}\\ \textit{coffee.}\end{array} :: \begin{array}{l}\textit{May I}\\ \textit{ask for}\\ \textit{two cups}\\ \textit{of tea?}\end{array} : x \quad \Rightarrow \quad x = \begin{array}{l}\textit{Two cups}\\ \textit{of strong}\\ \textit{tea.}\end{array}$$

In the word analogy task [11], [12], [13], especially for semantic or world knowledge categories, e.g., $\textit{brother} : \textit{sister} :: \textit{grandson} : \textit{granddaughter}$ or $\textit{Athens} : \textit{Greece} :: \textit{Oslo} : \textit{Norway}$ [1], the vector offset method is the method of choice to solve analogical equations in word embedding spaces where words are represented as dense vectors.[2] Various studies [14], [15] concluded that the distributed representations of quadruples satisfying some linear relationships embed the properties of analogy. Based on this findings, the potential of analogy is shown by its use as a benchmark to evaluate the quality of word embedding methods [16].

Compared with word analogies, analogies at the sentence level are confronted with more difficult challenges because of the structural compositionality of sentences and the higher semantic complexity. Following the success of word vector representations, sentence embeddings have received attention. They embed sentences into vectors to encapsulate semantic relations. A simple and effective way to represent variable-length sentences is to perform arithmetic operations (like averaging, summation, etc) on (sub)word vector representations [17], [18]. Different from these parameter-free methods, some approaches take sentence encoding as a part of their models, and train or fine-tune for specific objectives like contextual sentence prediction [19], natural language inference [20] and discourse marker prediction [21].

However, the lack of vector decoders limits research on sentence analogies to identifying the answer from a known set of candidates. It is also questionable whether simple linear operations are suited for solving analogies between sentences. Our contributions to remove these limitations are:

- We pre-train decoder networks with the ability to transfer sentence vectors independently of embedding methods, back into sequences of words.
- We introduce a network-based solution for sentence analogies which frees us from choosing among a set of candidates and which also performs better than the widely used approach based on linear operations.

II. Related Work

Sentence analogy is the task of deriving a sentence D that satisfies the relations in a given analogical equation $A : B :: C : D,$ where A, B and C are known sentences, and

[1]from the Google analogy data set. http://download.tensorflow.org/data/questions-words.txt

[2]https://aclweb.org/aclwiki/Analogy_(State_of_the_art)#Methods_to_solve_analogies

D is unknown. In comparison with word analogies, sentence analogies cover more complicated cases due to the intrinsic richness of sentences in lexical, syntactic and semantic variations and the differences in lengths. Intuitively, the probability of finding analogies between sentences is much lesser than between words. Early research like [22], [23] tackled this problem formally by treating sentences as strings of words or characters. They analyzed explicit literal relations without taking the meaning into consideration. The commutations of pieces in the sentences helped to generate solutions that meet analogical relations with the given three other sentences.

With the use of distributed representations, the linear vector offset method has yielded relatively high performance in solving both syntactic and semantic analogies between words, represented by fixed-size vectors [14]. Let us denote with $v(x)$ the embedding vector of a textual element x. The success of this linear method can be attributed to the regularity of vector offsets between entities in an analogy, mathematically expressed as $v(B) - v(A) \approx v(D) - v(C)$. The solution of an analogy is found by selecting the word with the highest cosine similarity to $v(B) - v(A) + v(C)$, from a given candidate set Λ, as stated in Equation (1).

$$\arg\max_{D \in \Lambda} \cos\left(v(D), v(B) - v(A) + v(C)\right) \tag{1}$$

The semantico-formal resolution of analogies between sentences has been introduced in [24]. Based on edit traces[3] between sentences, the analogical equation is decomposed into a set of semantic sub-analogies between sets of words, which, each, expect a single word as a solution. All optimum solutions of the sub-analogies eventually construct the sentence solution for the sentence analogy.

Guu et al. (2019) [25] trained a neural editor to obtain the edit vector encoding transformations between two sentences, and applied it on a sentence to generate a new sentence conditioned on this edit vector. Following the notion that the same relations jointly hold in two pairs of analogous items [1], they apply the edit vector representing the analogical relation in the sentence pair (A, B) on the sentence C to generate the answer D. In experiments of one-word sentence analogies, the ratios of which consist of two sentences with only one word difference, the accurate answers appear among the top ten outputs in about 55% of the cases. However, in most cases, it fails to directly generate the exact answer.

In recent years, fixed-size vector representation has attracted increased attention for the encoding of semantic and syntactic regularities of variable-length sentences. The quality of various sentence embedding methods capturing the linguistic properties of sentences have been investigated in [26]. On this basis, it is natural to extend the application of the linear arithmetic methods on sentence vectors, and identify the sentence with the maximum cosine similarity from a candidate vector. Diallo et al. (2019) [9] experimented on a sentence analogy data set composed of question-answer pairs. To enhance the quality of the answer selected from a set of candidates, they proposed a novel sentence embedding method to encode the analogical properties of quadruplets. Sentence vectors are taken as intermediate values of a Siamese network, trained to minimize the difference between two analogous pairs in the embedding space.

[3]A trace between two strings A and B is the representation of edit operations for transforming string A into string B.

III. Methodology

With the lack of a decoding method for sentence embeddings, selecting analogy solutions based on sentence vectors is limited to selecting the answer with the maximum cosine similarity against the hypothetical vectors of candidate sentences. To settle this issue, we introduce a decoder to reconstruct sentences encoded by vectors in a specific sentence representation space. Furthermore, we propose a neural network to predict the vector representing a sentence which meets the analogical constraints of an analogical equation. Subsequently, we use the pre-trained decoder to transform vectors into sentences.

A. *Decoder for Sentence Vectors*

We generate sentences on the premise of having only a fixed-size vector and no other information. For that, we train a vector-to-sequence model. Our approach is inspired by training a sequence-to-sequence (seq2seq) model [27] for monolingual translation. That is, we assume an auto-encoder architecture dedicated to reconstructing a sentence identical to the input sentence. The bottleneck layer contains the sentence vector generated by the conditional embedding model. The bottleneck is equivalent to the context vector given by the last hidden state of an RNN encoder in a seq2seq model. It represents the entire sentence. To learn the mapping between the vectors and the sentences in an embedding space, we fix the parameters of the pre-trained sentence encoder in order to preserve the property of the sentence embedding space. We focus on training a unidirectional RNN which predicts each token of the sentence from the given vector.

Our model is similar to the decoder part of *RNN Encoder-Decoder* in [28]. Given a sequence $S = (w_1, ..., w_N)$ of length N including a start-of-sequence token and an end-of-sequence token, the context vector c, which is the K-dimensional embedding of sentence S, is passed into the recurrent units as well as the softmax layer (classifier) at each time-step t, so as to retain the embedding of the complete sequence while updating the hidden state h_t and generating the next token w_{t+1}. In [28], the sentence vector c contributes to learning the hidden state and classifying the target word at each time step.

We concatenate the input token's embedding $v(w_t)$ and c with the output o_t of the RNN layer to predict the probability distribution of the token w_{t+1} against the words in the vocabulary set.

$$p(w_{t+1} | w_t, w_{t-1}, ..., w_1, c) = g(v(w_t), c, o_t) \tag{2}$$

In Equation (2), g is the softmax function. We call our decoder model ConRNN. Here, we use pre-trained vector models for the computation of the representation of sentences and of the expected tokens at every decoding step. We propose a multi-loss ConRNN model (ML-ConRNN) for the propose of restoring the correct order of words and a semantically similar text as the input. The multi-loss is defined as the combination of classification loss and regression loss. These two losses are explained below.

1) Classification Loss: As in previous work on seq2seq models, the goal is to make the predicted probability distribution close to the reference. We use a cross-entropy loss function to perform multi-label text classification given the sentence embedding c.

$$L_c = -\frac{1}{N}\sum_{n=1}^{N} \log p\left(w_n|c\right) \tag{3}$$

2) Regression Loss: This function deals with the similarity between the output of the recurrent units and the embeddings for the expected token at every time step. Mathematically, if the texts are semantically similar, the vectors are close. We expect that the output is very close to the embeddings of the next input words. Therefore, we apply the mean-squared-error (MSE) loss function to regress the last hidden states of RNN cells onto the next input embeddings, so as to obtain semantically meaningful words. This will hopefully preserve the information with respect to the order in the sequence through continuous regression.

$$L_r = \frac{1}{(N-1)\times K}\sum_{n=1}^{N-1}\sum_{k=1}^{K}(v(w_{n+1})_k - o_{nk})^2 \tag{4}$$

B. Resolution of Sentence Analogies

Our goal is to generate the solution (i.e., a sentence) of an analogical equation between sentences. We can compute the vectors of the known sentences A, B, and C, and we can transform a vector back to a sentence using the pre-trained decoder described above.

In this section, we introduce a neural network to learn the mapping between a given triple of sentences and the correct answer of the analogical equation in the embedding space, so that to predict the vector of the solution to an analogical equation. We then take advantage of our pre-trained decoder to decode the solution vector into a sentence which constitutes the solution of the analogical equation between sentences.

To predict the vector corresponding to the solution of an analogical equation between sentences, we establish a linear regression framework to learn the mapping between the composition of three known sentences and the embeddings of the expected solution. We experiment with three compositional methods on the known vectors $v(A)$, $v(B)$, $v(C)$:

- concatenation: $v(A) \cdot v(B) \cdot v(C)$
- summation: $v(A) + v(B) + v(C)$
- arithmetic analogy: $v(B) - v(A) + v(C)$

The number of neurons in the input and output layer corresponds to the size of the compositional vector and the sentence embedding, respectively. Since we want the generated vectors to be as close as possible to the correct solution of the sentence analogy in the embedding space, we use the MSE loss function of train the regression network. The function is given by Equation (5).

$$L_{MSE} = \frac{1}{K}\sum_{k=1}^{K}(v_k - v(D)_k)^2 \tag{5}$$

In Equation (5), v and $v(D)$ are the predicted vectors and the vectors of the labeled sentence respectively, and K is the dimension of the sentence embeddings.

TABLE I
THE STATISTICS OF DATA SET FOR TRAINING DECODERS

Data	Number of		
	sentences	words/sent.	characters/sent.
Training	63,336	6.7 ± 1.6	28.5 ± 8.0
Validation	7,917	6.7 ± 1.6	28.4 ± 8.0
Testing	7,918	6.7 ± 1.6	28.5 ± 8.0
Total	79,171		

TABLE II
THE STATISTICS OF DATA SET FOR TRAINING THE NETWORK WITH THE POTENTIAL TO SOLVE SENTENCE ANALOGIES.

Data	Number of			
	analogies	sentences	words/sent.	characters/sent.
Training	3,364	3,185	7.1 ± 1.2	27.0 ± 5.7
Validation	1,121	1,769	7.1 ± 1.1	26.6 ± 5.6
Testing	1,121	1,667	7.0 ± 1.1	26.3 ± 5.6
Total	5,607			

IV. Experiments

A. Data

We train the decoder on English sentences from the English-French parallel corpus Tatoeba[4]. Table I provides some statistics. The data set consists of 79,171 different sentences with an average length of 7 words (29 in characters). The words are 6 characters long in average. The length of the longest sentence is 10 words. We extract all the word types from the training set to build a vocabulary of 10,381 words.

In order to evaluate our proposal for the resolution of sentence analogies, we use the semantico-formal analogy set released in [24][5]. It comprises 5,607 labeled analogical equations between sentences extracted from the Tatoeba corpus, which include formal and semantic analogies between chunks. We split into 80%, 10%, 10% for training, validation and testing (see Table II).

B. Training Details

Long short term memory (LSTM) are known to perform better than gated recurrent unit (GRU). We build our RNN models using LSTMs. During the training process, we use a batch size of 128 to train our model and set the learning rate as 0.001 to tune the parameters in Adam optimizer. To avoid

[4]https://tatoeba.org/
[5]http://lepage-lab.ips.waseda.ac.jp/en/projects/kakenhi-kiban-c-18k11447/ See tab: Experimental Results

overfitting, we apply early stopping with a patience of 50. If the performance of the validation data set stops improving after waiting for an additional 50 epochs, we automatically interrupt it to obtain an optimal model.

When training the decoder, we use a pre-trained embedding model to embed sentences. The input size is the same as the vector size (768 dimensions for SBERT, and 300 dimensions for the composition of fastText vectors). At each decoding step, we apply teacher forcing, i.e., the network is explicitly taught by the reference with a random probability of 75%.

The network for solving sentence analogies comprises four linear layers, each hidden layer has 512 neurons. At each layer, we use the Leaky Rectified Linear Unit (LeakyReLU) as the activation function.

C. Evaluation Metrics

We assess the inferred results on several evaluation metrics, that are commonly used in text generation: BLEU score [29], METEOR Universal [30], and accuracy. The accuracy is defined as the number of predicted sentences which exactly match their reference divided by the total number of samples in the test set. It is measured by Formula (6).

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Number of samples in test set}} \times 100 \quad (6)$$

We evaluate use Jaccard similarity and Levenshtein distance between sentences in words and characters, in order to find how different two sentences are literally. To compute the Jaccard similarity, which is irrespective of the order of tokens, the sentence is regarded as a multi-set of words. If S_A and S_B are the notation for the multi-sets of A and B, respectively, the Jaccard similarity coefficient between the two multi-sets is computed like[6]:

$$J(S_A, S_B) = \frac{|S_A \cap S_B|}{|S_A \cup S_B|} \quad (7)$$

D. Performance of the Decoder

We evaluate the decoder models with the Nearest Neighbour method[7]: we retrieve the closest different sentence from the test set. To explore the quality of decoders against sentence embeddings, we perform experiments on two typical methods for representing variable-length sentences into fixed-size vectors: summation over word vectors and pre-trained sentence embedding model. We choose fastText[8] and SBERT[9] as our pre-trained models to obtain encoded vectors with dimensionality of 300 and 768 respectively. The results of decoding sentence vectors are reported in Table III[10].

[6]Given the multisets S_A and S_B, $S_A \cap S_B$ is the smallest multi-set that contains word(piece)s with the minimal number of occurrences in S_A and S_B. $S_A \cup S_B$ is the largest multi-set with all word(piece)s with the maximal number of occurrences in S_A and S_B.

[7]In order to find the nearest neighbour to the sentence S in the embedding space, from the candidate set $\{C_i\}$, the result should satisfy: $\arg\max_{C_i \neq S} \cos\left(v\left(C_i\right), v\left(S\right)\right)$.

[8]https://fasttext.cc/docs/en/crawl-vectors.html

[9]https://github.com/UKPLab/sentence-transformers

[10]We performed all experiments with GRU too and confirmed the advantage of multi-loss. Not surprisingly, the performance with GRU falls behind that of LSTM, without being statistically different.

The results show that RNN-based decoders perform significantly better than retrieving the nearest neighbour sentence from a pre-defined set. The BLEU score is multiplied by at least three. The worse performance of the closest sentences indicates that sentences with a similar semantic meaning in the vector space differ greatly in form. In other words, it is difficult to find a sentence similar, in form and meaning, to candidates composed of test sentences other than the references. For the offset method followed by nearest neighbour retrieval to be efficient, a necessary condition is that each analogical question be given a reasonable set of candidate sentences.

As shown in Table III, the ConRNN model with multi-loss outperforms all models in the two embedding spaces. Compared with the basic RNN trained with a single loss function, the accuracy of ML_ConRNN increases by more than 10%. By remembering sentence vectors through the addition of the regression loss, our model is capable of returning a semantic representation of the original sentences. In terms of accuracy, however, valid results which differ in form but not in meaning with the references are not considered accurate by the metrics we use. In addition, the limitation in vocabulary may negatively impact accuracy, which prevents the model to decode into unknown words.

We inspected the properties and the dimensions of the sentence embeddings used. The first group (summation over fastText vectors) achieves better results, although the sentence representations based on word vectors fail to encode the order of words. Because SBERT is trained with the goal of being able to compare embeddings with cosine similarity, using SBERT, the closest sentence should possess the highest semantic similarity. We list some examples of nearest neighbours against two embedding methods in Table IV. Although the values demonstrate that summation over word vectors is better, from the human point of view, some results obtained using SBERT's nearest sentences exhibit a closer meaning.

E. Results for Sentence Analogies

Our aim is to generate the sentences which are solutions of analogical equations by decoding them from candidate vectors representing the unknown in the equations. We first use the selected decoder to transform the embedding vectors of reference sentences D in the test analogy set (see Table V) and evaluate the performance on the analogical corpus. The decoder achieves high performance on the test set: 98% of the sentence vectors are perfectly decoded; on average, the generated sentences differ from the reference sentences by a fifth of a character; the Jaccard similarity is approximately 1.

Table VI presents the results obtained with different compositions of the embeddings of the known three sentences. We also report the performance of the vector offset method (see Equation (1)) which is a common way of solving analogies using embedding vectors. More specifically, we apply the arithmetic vector operation on the known vectors in the analogies, and then decode the resulting vector using the identical pre-trained decoder.

TABLE III
PERFORMANCE OF VARIOUS DECODING METHODS WITH DIFFERENT ARCHITECTURES TRAINED ON DIFFERENT SENTENCE EMBEDDING METHODS. THE FIRST GROUP CORRESPONDS TO THE SUMMATION OVER FASTTEXT VECTORS. THE SECOND GROUP IS THE EMBEDDINGS DERIVED FROM THE PRE-TRAINED SBERT MODEL "BERT-BASE-NLI-MEAN-TOKENS". FOR RNN-BASED MODELS, WE COMPARE TWO TYPES OF RNN, RECURRENT UNITS AND TWO DIFFERENT LOSS FUNCTIONS (SL FOR THE SINGLE LOSS (CLASSIFICATION LOSS), ML FOR THE MULTI-LOSS).

Decoding Method		**Edit Distance**		**Jaccard Similarity**		**Accuracy (%)**	**BLEU**	**METEOR**
Network	**Loss**	**in words**	**in characters**	**in words**	**in characters**			
Summation over fastText vectors (300 dim.)								
Nearest Neighbour		5.8 ± 2.8	20.5 ± 9.7	0.45 ± 0.15	0.60 ± 0.11	0.00	0.18 ± 0.01	0.25
basic RNN	SL	1.9 ± 2.5	7.6 ± 10.1	0.86 ± 0.19	0.89 ± 0.15	51.52	0.67 ± 0.01	0.48
	ML	1.4 ± 2.3	6.0 ± 9.6	0.92 ± 0.15	**0.94 ± 0.12**	62.38	0.74 ± 0.01	**0.53**
ConRNN	SL	1.6 ± 2.3	6.5 ± 9.3	0.89 ± 0.16	0.91 ± 0.14	56.57	0.72 ± 0.01	0.51
	ML	**1.3 ± 2.0**	**5.5 ± 8.8**	**0.93 ± 0.14**	**0.94 ± 0.12**	**62.84**	**0.76 ± 0.01**	**0.53**
SBERT (768 dim.)								
Nearest Neighbour		7.6 ± 3.1	23.7 ± 10.1	0.30 ± 0.17	0.54 ± 0.12	0.00	0.12 ± 0.01	0.20
basic RNN	SL	3.5 ± 3.2	11.9 ± 11.1	0.67 ± 0.26	0.77 ± 0.17	25.92	0.45 ± 0.01	0.38
	ML	3.2 ± 3.2	11.0 ± 11.1	0.70 ± 0.25	0.79 ± 0.17	30.00	0.50 ± 0.01	0.40
ConRNN	SL	3.2 ± 3.0	11.0 ± 10.8	0.70 ± 0.25	0.79 ± 0.17	29.14	0.50 ± 0.01	0.40
	ML	3.1 ± 3.0	11.0 ± 10.9	0.70 ± 0.25	0.79 ± 0.17	30.29	0.50 ± 0.01	0.41

TABLE IV
EXAMPLE RESULTS FOR NEAREST NEIGHBOUR WITH TWO DIFFERENT SENTENCE EMBEDDING METHODS.

Sentence	**Nearest neighbour & BLEU**				**Consistency with human judgment**
	Summation over fastText vectors		**SBERT**		
i never even considered that .	*i 'll never be that famous .*	0.01	*i never said that i wanted that .*	0.16	Yes
it can be difficult .	*it could be anybody .*	0.14	*it 's kind of complicated .*	0.01	No
we 're very grateful .	*we 're very busy .*	0.43	*i really appreciate this .*	0.11	No

TABLE V
RESULTS FOR DECODING THE VECTORS OF THE FOURTH TERM IN THE SENTENCE ANALOGIES USING THE PRE-TRAINED DECODER MODEL WITH THE BEST PERFORMANCE FROM THE FIRST GROUP (SUMMATION OVER FASTTEXT VECTORS) IN TABLE III.

Decoder Model	**Edit Distance**		**Jaccard Similarity**		**Accuracy (%)**	**BLEU**	**METEOR**
	in words	**in characters**	**in words**	**in characters**			
ML-ConRNN	0.1 ± 0.4	0.2 ± 1.7	1.00 ± 0.03	1.00 ± 0.02	97.59	0.99 ± 0.01	0.78

The vector offset method is largely used in word analogies. In our task, however, the arithmetic method faces the challenge of more complex relations encapsulated in sentences. The predicted sentences differ from the reference by one word in average. Recall that the upper limit of accuracy of the pre-trained decoder is around 98%, while, the accuracy of the vector offset method is only 42%, i.e., less than half of the upper limit. This indicates that the vector offsets are not kept within sentence embeddings.

The neural-based resolution method outperforms all models, except the model trained from summation of vectors. The vector size may explain this phenomenon. The results of concatenation, where three times larger vectors are used, show that larger representations encapsulate more information. However, we note that the performance of arithmetic analogy is superior to the other two methods even with the same size as summation. A BLEU score of 0.91 is obtained with a less than half a word. Table VII shows some results in the resolution of sentence analogies. Analogies where the edit distance between any two of the known sentences is less than half of their lengths, are more prone to be solved by our method. Also, analogies that include unknown words are difficult to solve.

V. CONCLUSION

We introduced decoder models for decoding sentence vectors into sentences. We showed that decoders trained for multi-loss objectives perform better. Based on this result, we proposed a linear architecture to learn the mapping between the vectors of the sentences present in an analogical equation and the vector of the sentence solution of this equation. We used the pre-trained decoder models to transform a solution vector into a solution sentence. By generating the answer sentence for a sentence analogy problem, we avoid the requirement of choosing among a set of candidates given in advance.

As the examples presented in Table IV show, our metrics measure well the formal features between sentences, but do not reflect so well the semantic similarity. To address this problem, the use of the metric recently proposed in [31] could be added. As for the decoder model, the introduction of conditional reward may lead to better generation of semantically meaningful tokens in the decoding steps.

ACKNOWLEDGMENT

This work was supported by grant number 18K11447 from the Japanese Society for the Promotion of Science (JSPS) enti-

TABLE VI
PERFORMANCE OF VARIOUS RESOLUTION METHODS TRAINED USING DIFFERENT COMBINATIONS OF TRIPLES OF VECTORS. FOR THE VECTOR OFFSET METHOD, WE APPLY THE SAME PRE-TRAINED MODEL ON DECODING THE OUTPUT VECTORS.

Resolution		**Edit Distance**		**Jaccard Similarity**		**Accuracy (%)**	**BLEU**	**METEOR**
	Composition	**in words**	**in char.**	**in words**	**in characters**			
Vector offset method		1.3 ± 1.2	5.0 ± 5.2	0.84 ± 0.14	0.85 ± 0.15	41.90	0.75 ± 0.01	0.50
Linear regression	concatenation	0.7 ± 1.4	2.5 ± 5.1	0.92 ± 0.15	0.93 ± 0.13	74.24	0.87 ± 0.02	0.59
	summation	1.6 ± 2.2	5.2 ± 6.9	0.82 ± 0.22	0.85 ± 0.18	52.41	0.72 ± 0.02	0.49
	arithmetic nlg.	**0.4 ± 1.1**	**1.6 ± 4.3**	**0.95 ± 0.12**	**0.96 ± 0.10**	**83.24**	**0.91 ± 0.01**	**0.64**

TABLE VII
EXAMPLES OF THE RESULTS OF SENTENCE ANALOGIES AGAINST DIFFERENT EDIT DISTANCES.

Sentence Analogy							**Prediction**	**Edit Dist.**	**BLEU**
i 'd like to be your friend .	:	*i just want to be your friend .*	::	*i 'd like to be alone .*	:	*i just want to be alone .*	*i just want to be alone .*	0	1.00
you were watching , were not you ?	:	*you were pretending , were not you ?*	::	*you were scared , were not you ?*	:	*you were **afraid** , were not you ?*	*you were **terrified** , were not you ?*	2	0.60
i 'm still a little hungry .	:	*i 'm a little sick .*	::	*i 'm still hungry .*	:	*i **am tired** .*	*i **'m sick** .*	4	0.19
her novel has been translated into japanese .	:	*his novel has been translated into japanese .*	::	*her novel was translated into japanese .*	:	*his **novel** was **translated into japanese** .*	*his **family** was **not quickly** .*	7	0.01
there is an urgent need for a new system .	:	*there is an urgent need for blood donations .*	::	*there is an urgent need for experienced pilots .*	:	***there is** an urgent **need for volunteers** .*	***no need need me involved** an urgent .*	10	0.13

tled "Self-explainable and fast-to-train example-based machine translation using neural networks".

REFERENCES

[1] D. Gentner, "Structure-mapping: A theoretical framework for analogy," *Cognitive science*, vol. 7, no. 2, pp. 155–170, 1983.

[2] D. Gentner and K. D. Forbus, "Computational models of analogy," *Wiley interdisciplinary reviews: cog. sci.*, vol. 2, no. 3, pp. 266–276, 2011.

[3] D. G. K. J. H. Boicho and N. Kokinov, *The analogical mind: Perspectives from cognitive science*. MIT press, 2001.

[4] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin, "Image analogies," in *CGIT*, 2001, pp. 327–340.

[5] M. Pirlot, H. Prade, and G. Richard, "Completing preferences by means of analogical proportions," in *MDAI*, 2016, pp. 135–147.

[6] Y. Lepage and E. Denoual, "The 'purest' EBMT system ever built: no variables, no templates, no training, examples, just examples, only examples," in *MT Summit X*, 2005, pp. 81–90.

[7] P. Langlais, F. Yvon, and P. Zweigenbaum, "Analogical translation of medical words in diff. languages," in *LNAI 5221*, 2008, pp. 284–295.

[8] H. Somers, S. Dandapat, and S. K. Naskar, "A review of EBMT using proportional analogies," in *EBMT III*, 2009, pp. 53–60.

[9] A. Diallo, M. Zopf, and J. Fürnkranz, "Learning analogy-preserving sent. embeddings for answer selection," in *CoNLL*, 2019, pp. 910–919.

[10] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *NIPS*, 2013, pp. 3111–3119.

[11] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *CoRR*, 2013. [Online]. Available: https://arxiv.org/pdf/1301.3781.pdf

[12] A. Drozd, A. Gladkova, and S. Matsuoka, "Word embeddings, analogies, and machine learning: Beyond king - man + woman = queen," in *COLING*, 2016, pp. 3519–3530.

[13] A. Rogers, A. Drozd, and B. Li, "The (too many) problems of analogical reasoning with word vectors," in **SEM*, 2017, pp. 135–148.

[14] T. Mikolov, W.-t. Yih, and G. Zweig, "Linguistic regularities in continuous space word representations," in *NAACL*, 2013, pp. 746–751.

[15] K. Ethayarajh, D. Duvenaud, and G. Hirst, "Towards understanding linear word analogies," in *ACL*, 2019, pp. 3253–3262.

[16] T. Schnabel, I. Labutov, D. Mimno, and T. Joachims, "Evaluation methods for unsupervised word embeddings," in *EMNLP*, 2015, pp. 298–307.

[17] J. Mitchell and M. Lapata, "Composition in distributional models of semantics," *Cogninitive Sciene*, vol. 34, pp. 1388–1429, 2010.

[18] A. Rogers, A. Drozd, and A. Rumshinsky, "Distributional compositional semantics in the age of word embeddings: tasks, resources and methodology," in *LREC*, 2018.

[19] R. Kiros, Y. Zhu, R. R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, and S. Fidler, "Skip-thought vectors," in *Advances in neural information processing systems*, 2015, pp. 3294–3302.

[20] A. Conneau, D. Kiela, H. Schwenk, L. Barrault, and A. Bordes, "Supervised learning of universal sentence representations from natural language inference data," in *EMNLP*, 2017, pp. 670–680.

[21] A. Nie, E. Bennett, and N. Goodman, "DisSent: Learning sentence representations from explicit discourse relations," in *ACL*, 2019, pp. 4497–4510.

[22] M. Nagao, "A framework of a mechanical translation between japanese and english by analogy principle," *Artificial and human intelligence*, pp. 351–354, 1984.

[23] Y. Lepage and G. Peralta, "Using paradigm tables to generate new utterances similar to those existing in linguistic resources," in *LREC'04*, 2004.

[24] Y. Lepage, "Semantico-formal resolution of analogies between sentences," in *LTC*, 2019, pp. 57–61.

[25] K. Guu, T. B. Hashimoto, Y. Oren, and P. Liang, "Generating sentences by editing prototypes," *TACL*, vol. 6, pp. 437–450, 2018.

[26] K. Krasnowska-Kieraś and A. Wróblewska, "Empirical linguistic study of sentence embeddings," in *ACL*, 2019, pp. 5729–5739.

[27] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *NIPS*, 2014, pp. 3104–3112.

[28] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *EMNLP*, 2014, pp. 1724–1734.

[29] M. Post, "A call for clarity in reporting BLEU scores," in *WMT*, 2018, pp. 186–191.

[30] M. Denkowski and A. Lavie, "Meteor universal: Language specific translation evaluation for any target language," in *EACL*, 2014.

[31] T. Sellam, D. Das, and A. Parikh, "BLEURT: Learning robust metrics for text generation," in *ACL*, 2020, pp. 7881–7892.