Coupling a Bi-CNN Encoder and a CNN decoder to automatically simplify English Sentences

Tim Patzelt, Dominik Pfuetze, Henny Sluyter-Gaethje, Simon Untergasser

Potsdam University
Matr.-Nr. 794948, 794948, 794380
{tpatzelt,dpfuetze,untergasser,sluytergaeth}@uni-potsdam.de

Abstract. Text simplification is the act of reducing the linguistic complexity of a text while preserving the information expressed in the text. Many approaches to perform this task automatically rely on the concept of neural machine translation. In our work we investigate whether having both a complex and a simple sentence as input for a convolutional neural network improves over networks having only one sentence as input.

Keywords: Text Simplification, Siamese Network, CNN

1 Introduction

To take part in society it is essential to have access to textual information about culture, literature, politics, economics, et cetera. It is even a fundamental right. Article 21a of the UN Convention on the rights of persons with disabilities affirms: States Parties shall take measures for "Providing information intended for the general public to persons with disabilities in accessible formats and technologies appropriate to different kinds of disabilities [...]" (UN, 2006). People diagnosed with dyslexia or alexia, who are marked by an impairment of the ability to recognize and comprehend words, could especially benefit of such measures. Some governments or organisations have translated documents into a language that is "easy-to-read" to assist persons with cognitive disability. This easy-to-read material is generally defined by the subsequent features: (a) The text is usually shorter than a standard text and redundant content is eliminated.[1] (b) Each sentence should only contain one piece of information avoiding subordinate clauses. (c) Previous knowledge is not taken for granted; backgrounds, difficult words and context is explained. (d) Easy-to-read is always easier than standard language, however there are differences of level depending on the target group in mind. For example, the *European Association of Persons with Intellectual Disabilities and their Families* has published the European standards for making information easy to read and understand (Inclusion-Europe, 2009). Nonetheless, most of the text material is not written in an easy-to-read for-

[1] Content reduction is controversial. For instance, the writing guidelines for the Simple English Wikipedia include the note "Simple English is not shorter English, although it frequently has shorter sentences".

mat. Since it would be costly and time-consuming to rewrite standard texts, research in Automatic Text Simplification (ATS) has developed for decades. But how can we simplify a complex sentence into an easy one "automatically"? There are three computationally straightforward ways of doing so: (1) Removing parts of the sentence; e.g. quotes, appositions, adjectives and adverbs, embedded clauses, attribution clauses. (Radev, 2016) (2) Substitute the Vocabulary; e.g. "she was condemned of drug trafficking" becomes "she was judged of drug trafficking". (3) Sentence breaking; e.g. "President Obama puts pressure on Republicans to accept a deal WHICH is aimed at avoiding a tax" becomes "President Obama puts pressure on Republicans to accept a deal. The deal is aimed at avoiding a tax." (Chanlekha, 2013). Applications can involve the sparsity of displays: e.g. for subtitling or on mobile devices. Another use can be written language in children's books or textual material for cognitive impaired people. Sentence simplification and segmentation can be performed by two approaches, either rule based (human created model) or corpus based (automatically created model from data). In our paper the corpus based technique is used for simplification. It is very easy to mix up automatic text simplification and automatic text summarization. ATS is different to text summarization as the focus of summarization is to reduce the length and the redundant content. However, ATS usually keeps all content, and the outputs are not necessarily shorter. Text simplification is also strongly related to but distinct from Machine Translation (MT), since we can regard the original English and the simplified English as two different languages. ATS has attracted much attention recently as it could make texts more accessible to wider audiences (Aluísio and Gasperin, 2010; Saggion et al., 2015). Besides it can be used as a preprocessing step to improve the performance of various NLP tasks and systems (Vickrey and Koller, 2008; Evans, 2011; Štajner and Popovic, 2016).

2 Related work

ATS research has been developed for decades. A good overview is given by Matthew Shardlow in "A survey of Automated Text Simplification". Shardlow is examining simplification research from the beginning of 1998 to 2013 (Shardlow, 2014). In the works before 1998 ATS is considered as a preprocessor for other NLP tasks such as parsing, summarization and machine translation (Chandrasekar et al., 1996). Then in (Carroll et al., 1998), an ATS system for aphasic readers is developed. The introduced system consists of a lexical tagger, a morphological analyser, a parser, a syntactic simplifier and a lexical simplifier (see Fig. 1, Carroll et al. (1998)). This architecture is already similar to most state-of-the-art simplification systems. The ongoing scientific series of evaluations of computational semantic analysis systems, SemEval, is also triggering research for ATS (McCarthy and Navigli, 2007; Specia et al., 2012). A quite unusual but interesting study by Yatskar et al. is utilizing edit histories of Simple English Wikipedia (SEW). They obtained 1.5M revisions of the SEW articles, producing many high-quality lexical substitution pairs (Yatskar et al., 2010). There are a lot of web based apps helping to get an intuition of ATS. The final result of a dissertation by Vasile Topac from the Polithenica University of Timisoara, Romania is a pop up dictionary for people with reading disabilities. In the web app one can open English and Romanian websites in a new window with a customized visual design and an included dictionary for difficult words, especially medical terminologies (Topac, 2014). Neil M. Goldman, a former computer scientist who is now an English teacher at a suburban high school in Chicago created a vocabulary simplifier, which he called Rewordify. His webpage rewordify.com shall motivate people to read more advanced literature than one would normally do by paraphrasing words in an abstract, providing statistics like complexity scores, lexical density, etc. and tag the parts of speech (Goldman, 2013). The Goodwill Company Ltd introduced Simplish, which uses an 850-words vocabulary based on Odgen's Basic English (Ogden, 1930) for fully simplifying text. The web tool also includes a scientific dictionary to help process scientific/technical text material (Funes-Gallanzi, 2014). The web pages csgenerator.com and spinbot.com are doing ATS the other way around. They both convert simple sentences into complex ones yielding a more pretentious language. The sentence "This is predicted to be awesome." becomes "This is presaged to be awe-inspiring." on one and "This is anticipated to be marvelous." on the other. The last four tools are especially used in search engine optimization (SEO) tasks, as search engines wants content to be unique and precise to get a better ranking. Now we dedicate ourselves to the current state-of-the-art automatic text simplification research for lexical and sentence simplification. One of the best phrase-based statistical machine translation (PBSMT) approach to ATS is using reranking of the n-best outputs according to their dissimilarity (Wubben et al., 2012). A mentionable ATS system simplifying the input sentence lexically and syntactically was introduced by researchers of the University of Aberdeen. The "conservative" lexico-syntactic ATS system does not perform any content reduction and thus completely preserves the original meaning of the sentence (Mandya et al., 2014). Xu et al. introduced one of the recently leading syntactic-based machine translation (SBMT) systems (Xu et al., 2016). They developed a new text-simplification metric (SARI) for their tuning function and used the PPDB paraphrase database (Ganitkevitch et al., 2013). All of the aforementioned ATS systems were surpassed by the work of Sergiu Nisioi and Sanja Stajner, who used a neural sequence to sequence (Seq2seq) model, also referred to as Encoder-decoder models. They ranked the output with different metrics and significantly outperformed current systems, shown by human evaluation. Their neural text simplification (NTS) model can jointly perform lexical simplification and is capable of correctly performing significant content reduction (Nisioi et al., 2017). We also abandon the idea of PBSMT, the conservative lexico-syntactic way and the SBMT approach and use a novel conception. A Bi-Convolutional Neural Network (Bi-CNN). It was introduced by Yin et al., achieving state-of-the-art performance on three NLP tasks: answer selection (AS), paraphrase identification (PI) and textual entailment (TE) (Yin et al., 2015). "Bi-CNN" stands for double CNNs used in a Siamese framework (Yin and Schütze, 2015). We loosely orient on the Bi-CNN architecture. That means our Encoder consists of two weight-sharing CNNs, one processing a simple sentence and the other a complex sentence, and a final layer which solves the sentence pairing task. The special part is our decoder architecture, which is built like a usual CNN. Our decoder processes the sentence pair of the Encoder and ends with a final layer decoding the pair into a simple sentence.

3 Data and Methods

3.1 The Wikipedia Corpus

We use the publicly available dataset of English Wikipedia and Simple English Wikipedia (EW-SEW). Yet, in a specialized version. Hwang et al. (2015) introduced a sentence alignment method using a new word-similarity measure and a greedy search over sentences and sentence fragments, resulting in good, good partial, partial or bad matches. Following Nisioi et al. (2017), we discard the partial and bad matches and only use good matches and good partial matches which were above the 0.45 threshold (Hwang et al., 2015). Thus the corpus consists of 284.678 aligned sentences, with about 150K good matches and about 130K good partial matches. Unlike previously used EW-SEW corpora which only contain full matches (Zhu et al., 2010; Woodsend and Lapata, 2011; Kauchak, 2013; Xu et al., 2016), it is one of the biggest freely available resource for ATS. The new dataset also includes good partial matches which additionally allows for learning sentence shortening, as shown in Table 1 (Nisioi et al., 2017).

3.2 Preprocessing / Cleaning

The preprocessing steps differ depending on whether we deploy the Bi-CNN or an End2End solution, where Encoder and decoder are joined. We wanted to use the latter to determine whether optimizing the Encoder and the Decoder separately improves the performance of the network.

In the Bi-CNN the goal of training the Encoder is to have similar representations of the simple and complex sentence that were aligned to each other. To achieve this we make use of negative sampling i.e. the representations of sentences that don't belong together should be least similar. We first oriented the ratio of positive and negative examples towards (Yin et al., 2015). Their dataset consisted of 4 incorrect answers to one correct answer. Since the performance of our Encoder was bad, we switched to a 1:1 ratio and also made sure that the negative examples were distant from one another. To determine the distance we used a simple comparison of the vocabulary of both sentences. We set the similarity threshold to one third, meaning that the sentences are distant enough if only one third or less of the vocabulary size are the same words. We labeled the positive examples with 1 and the negative examples with 0. Since in the End2End approach we did not train the Encoder separately we did not have to make use of labels to be able to update the weights in the Encoder. The first step in transforming words to vectors was to tokenize the sentences. As a part of the cleaning process we restricted the sentences in

	complex	*simple*
sentence shortening (drop of irrelevant information)	*Falaj irrigation is an ancient system dating back thousands of years and is used widely in Oman, the UAE, China, Iran and other countries.*	*The ancient falaj system of irrigation is still in use in some areas.*

Table 1. Example of a complex and the coreesponding simple sentence.

size as the matrices would have been too large otherwise. We achieved the best performance when restricting the sentences to 40 words respectively. We also deleted sentences for which a certain percentage of words was not part of our Word2Vec vocabulary. We set the threshold of unknown words to be ten percent of the length of the sentence. After the preprocessing and the cleaning, the corpus was reduced from 284.678 to 196.720 aligned sentences. As we used negative sampling the number of input sentences doubled to 393.440 sentences.

3.3 Word2Vec or FastText?

Initially, we used the pre-trained Google News corpus word vector model (3 million 300-dimension English word vectors) to transform our complex and simple sentences into word embeddings (Mikolov, 2013). Very soon, however, we noticed that many words in our corpus are unknown to the Google News word vector model. This is when we tried FastText. Word2Vec treats every word as the smallest unit to train on. FastText instead treats each word as composition of n-grams. The word vector "ball", for instance, is the sum of the vectors of the character n-grams "<ba", "bal", "ball", "all", "ll>"; with smallest n-grams set to 3 and highest to 4 (Huang, 2018). We decided to use FastText (Bojanowski et al., 2016), trained on the English Wikipedia[2]. We have set a threshold for unknown words. As you can see in Figure 1, if we set the threshold to 1% of words unknown to FastText, we already achieve over 40% (132.873 sentences) of remaining sentences from the original corpus (284.678 sentences). By comparison, we achieve the same only after setting the threshold to more than 30% unknown words to Word2Vec. We set final threshold for unknown words to 10% wich enabled us to use nearly 70% of the corpus.

3.4 Encoder

Our Encoder architecture is strongly inspired by the BCNN introduced by Yin et al. (2015). But instead of a classification task which is solved by logistic regression on the two final sentence representations we try to train the network such that both representations are similar. This means the information needed for the decoder to produce a simple English sentence are preserved in both representations. In the following section, the BCNN will be in-

[2] We took the pretrained vectors from https://github.com/facebookresearch/FastText/blob/master/pretrained-vectors.md

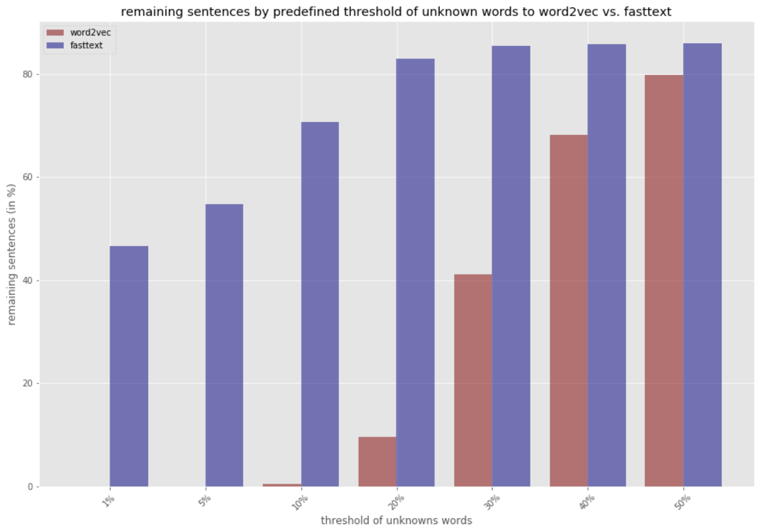

Fig. 1. unknown words to Word2Vec / FastText

troduced and our modifications to it are laid out.

3.4.1 BCNN The BCCN proposed by Yin et al. (2015) can be assigned to the class of Siamese Networks which contain at least two identical subnetworks meaning they share the same parameters and weights. They use two CNNs in parallel to process two input sentences followed by a final layer which connects the two sentence representations and solves a given task. In their paper, they present the results for tasks such as Answer Selection, Paraphrase Identification and Textual Entailment. The structure can be seen in Figure 2 in (Yin et al., 2015) and will be described shortly in the following section.

The input layer consists of the word embeddings computed by Word2Vec (Mikolov et al., 2013) or FastText (Bojanowski et al., 2016),where each feature vector of a sentence is of dimension $[d0, \ s]$ with $d0 = 300$ and $s = max(length(s1), length(s2))$.

In each convolution layer *wide convolution* is applied to the feature vectors such that the rightmost and leftmost words are detected by each convolution weight in W. This is achieved by padding the feature vectors with w-1 zeros and results in $s + w$ different concatenations for convolution with w beeing the window size of the convolution. Each concatenation of word embeddings can be seen as a phrase consisting of words to which convolution is applied in the following way

$$p_i = tanh(W * c_i + b)$$

where in W is the convolution weight matrix and b are the biases.

In all non-final convolution layers *w-average pooling* (w-ap) is applied and it is defined as column-wise averaging over w consecutive columns. Windows with w columns ensures that the dimension of the output of each layer is the same as the input dimensions because a convolution filter of width w generates for s columns $s + w - 1$ columns. Performing the *w-average pooling* transforms the columns back to size s.

In the final convolution layer *average pooling over all columns*(all-ap) is applied, generating a sentence representation with dimension 1 X 300 and thus "summarizing" the final feature map. Besides w-ap in each non-final layer, all-ap is also applied in the final layer. Starting from the first layer operating on words, the feature maps of the following layer represents more abstract feature which resemble properties potentially meaningful to the problem. The representation obtained by all-ap on the final convolution layer as well as the representations obtained by all-ap on all convolution layers are passed directly to a model which solves one of the aforementioned task. While testing out different numbers of layers Yin et al. (2015) report that they achieve the best results with two convolution layers and adding more does not add any value.

3.4.2 Modifications to the network

To use the BCCN as Encoder we had to make several adjustments to the network. An overview is given in Figure 2. First of all, we do not want to solve a task on the final sentence representations but rather take the output of the trained Encoder as input to a Decoder reconstructing a simple English sentence from it. Therefore, we cannot measure the performance on that task. Our idea was to train the Encoder in such a way that it learns the features which are distinctive for a simple sentence. In other words, we want a mapping from an embedding of a simple or complex sentence to a representation which corresponds to the embedding of the simple sentence. Because we have sentence pairs with corresponding simple and complex sentences as well as sentence pair with random pairs from the corpus, we had to label each sentence pair with either 1 meaning they correspond or 0 meaning they do not belong together. Furthermore, we decided to start with the euclidean distance as similarity measure because it is an intuitive measure of how similar two vectors are. The optimization criterion can thus be formulated as (1) where x_1 is the simple sentence and x_2 is the complex one.

While setting up the Encoder, we have already started training our end-to-end network (explained in the next section) with different parameters and observed that the end-to-end network does not learn anything yet. To improve it we switched from euclidean distance to cosine distance which is defined as

$$dist_{cos}(x, y) = \frac{x \cdot y}{||x|| \cdot ||y||}$$

and ranges from 1 (same angle) to 0 (orthogonal). The intuition is that it

$$crit = \begin{cases} maximize(sim_{eucl}(x_1, x_2) & if\ label(x_1, x_2) = 1 \\ minimize(sim_{eucl}(x_1, x_2) & if\ label(x_1, x_2) = 0 \end{cases} \tag{1}$$

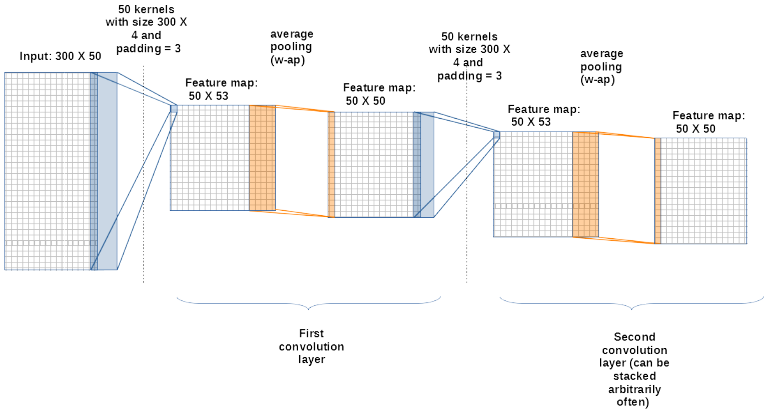

Fig. 2. Our modified BCNN with two convolution layers

$$crit = minimize((label(x_1, x_2) - sim_{cos}(x_1, x_2))^2) \tag{2}$$

works better in the high dimensional space of our embeddings ($d_0 = 300$) because it neglects the magnitude of the vectors but instead considers their orientation. For example, the vectors

$$x1 = \begin{pmatrix} 2 \\ 5 \end{pmatrix} \quad \text{and} \quad x2 = \begin{pmatrix} 4 \\ 10 \end{pmatrix}$$

have a euclidean distance of

$$dist = \sqrt{\begin{pmatrix} 2 \\ 5 \end{pmatrix}^2 + \begin{pmatrix} 4 \\ 10 \end{pmatrix}^2} \approx 5.4$$

but a cosine similarity of 1, meaning their angles are identical. Furthermore data points in high-dimensional spaces have roughly equal pairwise Euclidian distance (see for instance (Aggarwal et al., 2001)). Using the cosine similarity we could reformulate the loss function as (2). Here, we do not need to differentiate between two cases anymore because the cosine similarity reaches from 0 to 1 and hence can directly compared to the label of the sentence pair. The next modification we made was that we increased the number of convolution layers from two to four. We run training with models consisting of two, three and four layers and observed that four works the best there.

3.5 Decoder

The architecture of the Decoder mirrors the Encoder being inspired from the AutoEncoder idea. The overall goal was to have the decoder perform a deconvolution on the Encoder output. Since convolution, as we applied it, is a destructive operation in the sence that information about the magnitudes of the inputs is lost, it can't be reversed. Its an ongoing research question to find good algorithms to approximate the inputs of a convolution only by the outputs and the weights. The second problem is the pooling operation which comes after every layer. Average- or max-pooling also destroys information.

The available operation which mimics deconvolution the most is transposed convolution, sometimes also called fractionally strided convolution. The idea behind this operation is to reconstruct the spatial resolution of the input. So while not performing a deconvolution transposed convolution allows to reverse the spatial compression of the convolution operation. The hope for our project was, that the decoder is able to learn a proper "deconvolution".

3.5.1 Transposed Convolution

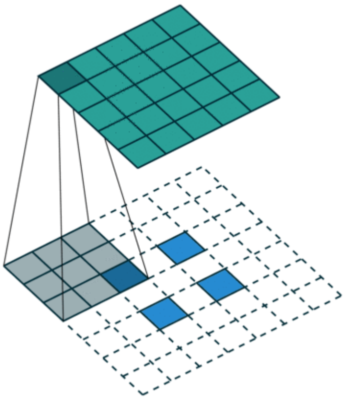

Fig. 3. Zero padded 2x2 input to produce an 5x5 output.
Image retrieved on 11.09.2018 from: http://deeplearning.net/software/theano /tutorial/conv_ arithmetic.html

This operation produces the same spatial resolution as a deconvolution would do. It achieves this by padding the input with zeros in a way, that performing normal convolution on this padded input produces an upscaled output. Figure 3 gives an impression of the operation. Given a 3x3 convolution kernel and a stride of 2 the convolution operation would go from the 5x5 layer on the top to the 2x2 output on the bottom. The transposed convolution on the contrary performs an operation from the bottom layer as input (padded with zeros) to the top layer.

3.6 End-to-End Network

As alternative to a separate training of the Encoder and decoder we tried to deploy a model which couples them into a single end-to-end model, sketched in Figure 4. Instead of optimizing the loss function for the Encoder and decoder sequentially, the performance is only evaluated on the final output. Because there is no intermediate optimization criterion anymore, i.e. the similarity of the output of the Encoder for simple and complex sentences, we decided to take only word embedding of complex sentences as input and compared it to the embedding of the corresponding simple sentence as gold standard. We started out by using the euclidean distance

$$dist_{eucl} = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

as distance measure for our word vectors as it is an intuitive measure of how close they are in vector space. After training the network for 1000 epochs several ties, we observed that the network does not learn anything. To improve our network we switched from euclidean distance to cosine distance, by the reasoning mentioned before. Nevertheless, our network did not improve and we decided to stop the

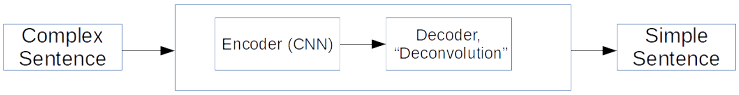

Fig. 4. A sketch of our end-to-end approach.

end-to-end approach because we found it very hard to gain insights into the network and debug it properly.

4 Results

Table 2 shows the results of the different models we used. In the beginning, we experimented with the Encoder using euclidean distance, but we were not able to reach more than 58.7% mean similarity on the test set. After redefining our loss function using the cosine distance, the mean similarity zoomed up to 97.57% using four convolutional layers.

Because the Decoder was trained after the Encoder, we directly used the cosine distance as loss function. Nevertheless, the mean similarity of the Decoder on the test set peaked at 36.7%. Interestingly, the similarity was the same for three and four layers, so we decided to report the results of the model with three.

As alternative to coupling the Encoder and the Decoder, the End-to-End model was trained with the same loss function as the Decoder but only reached a mean similarity of % on the test set. Therefore, one could say that the separate training of the Encoder

Model	distance func.	#Layers	mean similarity (train set)	mean similarity (test set)
Encoder	euclidean	4	60.02	58.7
Encoder	cosine	4	98.7	97.57
Decoder	cosine	3	36.61	36.7
End-to-End	euclidean	8	25.6	24.92
End-to-End	cosine	8	30.42	30.2

Table 2. The final results of each model we used.

sample output 1	sample output 2
éŇŸ éŇŸ éŇŸ éŇŸ éŇŸ éŇŸ éŇŸ éŇŸ éŇŸ éŇŸ éŇŸ éŇŸ éŇŸ éŇŸ éŇŸ áŻń áŻń éŇŸ éŇŸ áŻń áŻń áŻń áŻń áŻń áŻń áŻń áŻń áŻń áŻń æźŁ which which which which æźŁ æźŁ æźŁ æźŁ bouÃŕnian which garian kiraÃğ seicho megalynarionkiraÃğ mahÄĄrÄĄsháźŋra åăž ä¡ŢéťżçĞŁ musuh aÅč éŇŸ éŇŸ éŇŸ éŇŸ éŇŸ éŇŸ éŇŸ éŇŸ éŇŸ éŇŸ éŇŸ éŇŸ áŻń éŇŸ éŇŸ áŻń áŻń âŹĎ éŇŸ áŻń áŻń æźŁ æźŁ áŻń áŻń áŻń áijŢ áŻń áŻń æźŁ áŻń æźŁ áŻń æźŁ æźŁ æźŁ æźŁ æźŁ bouÃŕnian which garian kiraÃğ seicho megalynarion kiraÃğ mahÄĄrÄĄsh áźŋra åăž ä¡ŢéťżçĞŁ musuh aÅč	thážčm da_id thážčm cn/ pofi irÃş thážčm dosimo dosimo dosimo dosimo thážčm dosimo dosimo thážčm ØğÙĎØťØğØĺ âĘŘso àďňàěŊàďřàď£àď§àď£àďű #radioactive àšňàşĘàššàşŊàšš sequestra cryolathe interposing toglie stereotypes_of_white_people liisi thážčm dosimo dosimo dosimo dosimo dosimo dosimo dosimo dosimo dosimo dosimo dosimo thážčm thážčm

Table 3. Left is the translated output of the Siamese approach, on the right side the End-to-End one.

and Decoder yields better results than the End-to-End model.

4.1 From Word embeddings to Words

Because we were interested in how our output would look on word level, we thought about methods to retrieve the word form. Because of time issues, we decided to use the quite simple method of looking for the closest vector (nearest neighbor) in the word embedding space for which we know the source word. To find the closest neighbor, we again relied on the cosine similarity. Two sample outputs are translated and presented in Table 3. As one can see, the sentences contain a few English words and mainly Vietnamese, Chinese and Russian words.

5 Discussion

Our goal was to find out whether it is beneficial to use a sentence pair with a simple and complex sentence over using a single sentence. Both models had an Autoencoder structure, but one was trained End-to-End while in the other the Encoder and Decoder were trained separately. Overall, the results were not overwhelming. We wanted to transform the word embedding of a complex sentence to the one of the simple sentence. The best result was achieved by the separate Encoder-Decoder structure with a final mean similarity of 36.7% between the simple and complex sentences in the test set. A similarity of 36.7% can be translated int an orientation. The output of our model is on average 68.47° of the word embedding of the simple sentences. Considering that we do not look at the magnitude of the vectors at the moment, the results are not satisfying. Furthermore, translating word embeddings back to actual words is not a trivial task, which we underestimated in the beginning. We took a fairly simple approach including the cosine similarity which can be a very extreme measure. According to Beyer et al. (1999), data points tend to be pushed in the corners in high-dimensional spaces and the "middle" becomes empty or sparse. Having this in mind, cosine distance might not be able to differentiate well between different points anymore.

The last major conclusion which can be drawn from the results is that the data set is very noisy. Using only single sentences from the English, respective Simple English, Wikipedia the importance and use of context information is neglected completely. While the writers of the Simple English Wikipedia used information from the whole English article, our models could only access the information present in one sentence. Thus, resolution of co-references and adding of additional information could not happen this way. Moreover, this leads sometimes to simple sentences which exceed the length of 1.5 times the length of the complex one. A generally valid definition of simple is still not defined as addressed in the introduction. Finally, there were matches of sentences in the corpus which do not belong together at all, aggravating the learning of the models. To overcome these issues, different ideas and approaches are presented in the next section.

5.1 Ideas for improvement

As we ran out of time, we were not able to test a few more ideas we had. Furthermore we got inspiration from the discussion after the final talk. This section gives a brief overview of the ideas worth pursuing. One of the main difficulties is the translation from word vectors to words as discussed in the section before. There are several approaches which could improve results. The first idea is to train FastText on our own dataset. This would probably help to avoid unknown symbols as the ones in Table 3. When we would train FastText from scratch, we probably would also reduce the dimensions of the word vectors to 100. This reduction in complexity would allow for faster training and therefore more iterations of training. The reduced dimensions would also allow for stacking more layers. More convolutional layers increase the span over the sentence. Another big problem is the nearest-neighbor search mentioned in the last section. In these high dimensional vector-spaces similar words are grouped together. But very rare or uncommon words are scattered all over the space. So a query for a word by nearest-neighbour search has a high probability of resulting in one of these uncommon words. To avoid this problem we could discard all words which are rare. Another approach is to avoid the search in the vector space completely. We could insert a fully connected layer trained to give a probability distribution over all words. This would result in a learned mapping from word vectors to words. Another improvement concerns the data preprocessing. Since the corpus is very noisy as mentioned in the section before a simple similarity measure of complex and simple sentence could reveal, if they are good training examples. A lot of simple sentences are completely different then the corresponding complex ones. There is no relation between them, which could be learned by the network. The similarity could be realized by a bag-of-words comparison.

6 Summary

All in all, we showed that, at least to a small extend, using sentence pairs over single sentences leads to better results. The idea of training the Encoder in such a way that it creates the same representation for a simple and a complex one was achieved. Whether this representation contains the information needed to reconstruct the simple sentence still needs investigation, probably using the ideas we laid out before. First experiments with a fully connected layer showed already improvements of around 8% for the End-to-End model. In our opinion, implementing the other improvements, especially the transition from word embeddings to words, could lead to better results in the future.

Bibliography

Aggarwal, C. C., Hinneburg, A., and Keim, D. A. (2001). On the surprising behavior of distance metrics in high dimensional space. In *International conference on database theory*, pages 420–434. Springer.

Aluísio, S. M. and Gasperin, C. (2010). Fostering digital inclusion and accessibility: the porsimples project for simplification of portuguese texts. In *Proceedings of the NAACL HLT 2010 Young Investigators Workshop on Computational Approaches to Languages of the Americas*, pages 46–53. Association for Computational Linguistics.

Beyer, K., Goldstein, J., Ramakrishnan, R., and Shaft, U. (1999). When is âĂIJnearest neighborâĂİ meaningful? In *International conference on database theory*, pages 217–235. Springer.

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2016). Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Carroll, J., Minnen, G., Canning, Y., Devlin, S., and Tait, J. (1998). Practical simplification of english newspaper text to assist aphasic readers. In *Proceedings of the AAAI-98 Workshop on Integrating Artificial Intelligence and Assistive Technology*, pages 7–10.

Chandrasekar, R., Doran, C., and Srinivas, B. (1996). Motivations and methods for text simplification. In *Proceedings of the 16th conference on Computational linguistics-Volume 2*, pages 1041–1044. Association for Computational Linguistics.

Chanlekha, H. (2013). Read-up: Text simplification for reading comprehensions (android rss feed reader). *https://www.youtube.com/watch?v=li_-AVWXLMxw*.

Evans, R. J. (2011). Comparing methods for the syntactic simplification of sentences in information extraction. *Literary and linguistic computing*, 26(4):371–388.

Funes-Gallanzi, M. (2014). Simplish - an automated multilingual simplifying and summarizing tool. *https://simplish.org/*.

Ganitkevitch, J., Van Durme, B., and Callison-Burch, C. (2013). Ppdb: The paraphrase database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 758–764.

Goldman, N. M. (2013). Rewordify - understand what you read. *http://rewordify.com/*.

Huang, S. (2018). Word2vec and fasttext word embedding with gensim. *https://towardsdatascience.com/word-embedding-with-word2vec-and-fasttext-a209c1d3e12c*.

Hwang, W., Hajishirzi, H., Ostendorf, M., and Wu, W. (2015). Aligning sentences from standard wikipedia to simple wikipedia. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 211–217.

Inclusion-Europe (2009). Information for all: European standards for making information easy to read and understand. *www.easy-to-read.eu/wp-content/uploads/2014/12/EN_Information_for_all.pdf.*

Kauchak, D. (2013). Improving text simplification language modeling using unsimplified text data. In *Proceedings of the 51st annual meeting of the association for computational linguistics (volume 1: Long papers)*, volume 1, pages 1537–1546.

Mandya, A. A., Nomoto, T., and Siddharthan, A. (2014). Lexico-syntactic text simplification and compression with typed dependencies. In *25th International Conference on Computational Linguistics.*

McCarthy, D. and Navigli, R. (2007). Semeval-2007 task 10: English lexical substitution task. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 48–53. Association for Computational Linguistics.

Mikolov, T. (2013). Google code archive. *https://code.google.com/archive/p/word2vec/.*

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781.*

Nisioi, S., Štajner, S., Ponzetto, S. P., and Dinu, L. P. (2017). Exploring neural text simplification models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 85–91.

Ogden, C. K. (1930). Basic english: A general introduction with rules and grammar.

Radev, D. (2016). Lecture059 - sentence simplification. *https://www.youtube.com/watch?v=0X4zlwXujco.*

Saggion, H., Štajner, S., Bott, S., Mille, S., Rello, L., and Drndarevic, B. (2015). Making it simplext: Implementation and evaluation of a text simplification system for spanish. *ACM Transactions on Accessible Computing (TACCESS)*, 6(4):14.

Shardlow, M. (2014). A survey of automated text simplification. *International Journal of Advanced Computer Science and Applications*, 4(1):58–70.

Specia, L., Jauhar, S. K., and Mihalcea, R. (2012). Semeval-2012 task 1: English lexical simplification. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 347–355. Association for Computational Linguistics.

Štajner, S. and Popovic, M. (2016). Can text simplification help machine translation? In *Proceedings of the 19th Annual Conference of the European Association for Machine Translation*, pages 230–242.

Topac, V. (2014). text4all - making text and web pages more friendly for you. *http://www.text4all.net/.*

UN (2006). Convention on the rights of persons with disabilities. *www.un.org/disabilities/documents/convention/convoptprot-e.pdf.*

Vickrey, D. and Koller, D. (2008). Sentence simplification for semantic role labeling. *Proceedings of ACL-08: HLT*, pages 344–352.

Woodsend, K. and Lapata, M. (2011). Learning to simplify sentences with quasi-synchronous grammar and integer programming. In *Proceedings of the conference on empirical methods in natural language processing*, pages 409–420. Association for Computational Linguistics.

Wubben, S., Van Den Bosch, A., and Krahmer, E. (2012). Sentence simplification by monolingual machine translation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 1015–1024. Association for Computational Linguistics.

Xu, W., Napoles, C., Pavlick, E., Chen, Q., and Callison-Burch, C. (2016). Optimizing statistical machine translation for text simplification. *Transactions of the Association for Computational Linguistics*, 4:401–415.

Yatskar, M., Pang, B., Danescu-Niculescu-Mizil, C., and Lee, L. (2010). For the sake of simplicity: Unsupervised extraction of lexical simplifications from wikipedia. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 365–368. Association for Computational Linguistics.

Yin, W. and Schütze, H. (2015). Convolutional neural network for paraphrase identification. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 901–911.

Yin, W., Schütze, H., Xiang, B., and Zhou, B. (2015). Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *arXiv preprint arXiv:1512.05193*.

Zhu, Z., Bernhard, D., and Gurevych, I. (2010). A monolingual tree-based translation model for sentence simplification. In *Proceedings of the 23rd international conference on computational linguistics*, pages 1353–1361. Association for Computational Linguistics.