Finding Structure in Neural Network Activation Patterns via Representational Similarity and Convolutional Kernels

Dennis de Groot

ANR: 408280

THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE IN COMMUNICATION AND INFORMATION SCIENCES, MASTER TRACK DATA SCIENCE BUSINESS & GOVERNANCE, AT THE SCHOOL OF HUMANITIES AND DIGITAL SCIENCES OF TILBURG UNIVERSITY

Thesis committee:

dr. G.A. Chrupała

prof. dr. E.O. Postma

Tilburg University

School of Humanities and Digital Sciences

Department of Cognitive Science & Artificial Intelligence

Tilburg, The Netherlands

December 2018

Table of Contents

Abstract

The rise of neural network models in various domains has increased the interest in methods of analysing and interpreting the representations learned by these models. Specifically, in computational linguistics, sentence encoders are being evaluated on a detailed level, such as evaluating the hidden activation patterns of Recurrent Neural Networks (RNNs) or assessing performance on specific tasks. This study offers a global approach to quantify how well a neural representation space corresponds to a structured symbolic representations space. This approach was used to find to what extent neural representations capture the syntax of sentences as opposed to their semantics. By applying Representational Similarity Analysis to various representations of sentences from the novel The Little Prince by Antoine de Saint-Exupéry, it was found that neural representations capture the semantics to a larger extent than the syntax. Moreover, this novel approach provided a new perspective on the representations learned by RNNs. By combining differently encoded vectors, which seem to be biased to capturing either the semantics or the syntax of a sentence, fuller representations can be made which are less biased and capture both the semantics and syntax. In conclusion, this novel approach provides a more global evaluation of neural representations, as well as offering a new perspective from which these representations can be studied.

1. Introduction

As neural network models are being used for an increasingly wide variety of domains, interest in methods of analysing and interpreting the representations they learn have been growing. Specifically in computational linguistics, a number of approaches have been proposed for the purpose of correlating neural representations with symbolic structures from linguistic theory such as phonemes, phoneme sequences, syntactic tree and semantic representations. In this study we propose a simple approach to quantify how well a neural representation space corresponds to a structured symbolic representation space.

Recurrent neural networks (RNNs) were introduced by Elman (1990). They have the ability to model sequential data and can learn representations of linguistic units directly from input data. RNNs have been used increasingly for various NLP tasks, e.g. parsing (Vinyals et al., 2015; Dyer et al., 2016) and machine translation (Bahdanau, Cho, & Bengio, 2014). They transform linguistic expressions of variable lengths to a representation in the form of a fixed-size low-dimensional vector (Kádár, Chrupała, & Alishahi, 2017). This representation is a complex, non-linear function of the input which raises problems for interpretability, accountability and controllability of NLP systems. Therefore, interest has grown into making these representations explainable by investigating how they acquire abstract linguistic knowledge and to what extent they learn this.

Two approaches can be taken to study the neural representations which RNNs learn. The first approach focuses on examining the hidden activation patterns of RNNs (Karpathy, Johnson, & Fei-Fei, 2015). The second approach tries to understand these learned representations similar to how we study human language processing (Linzen, Dupoux, & Goldberg, 2016). This is done by assessing behaviour on targeted sentences to study the learned representation. The two approaches will be further explained in Section 2.

However, these two approaches are focussed on evaluating specific activations or performance on specific tasks. This study proposes a more general approach to quantify how well a neural representation corresponds to a structured symbolic representation. It provides a framework to evaluate and compare neural representations on a more global level, enabling easier analysis of the learned syntax and semantics. Therefore, it offers a new perspective on the neural representations learned by RNNs.

This study contributes to the scientific search for insights into representations learned by neural networks by providing a framework in which models can be easily evaluated on learned

syntax and semantics. This enables more general evaluations and comparisons between neural networks instead of evaluation on smaller, specific tasks. As neural networks are increasingly being used for practical means as well, a better understanding of the representations they learn can lead to advancements for societal applications as well.

The main research question in this thesis is as follows: “To what extent does a neural network representation capture the syntax of sentences as opposed to the semantics?” In this study, the neural network representation of focus is constructed using the skip-thoughts model provided by Kiros et al. (2015). Rationale as to why this model is chosen will be explained in Section 4. One advantage of using this model is that it produces a vector, which is combined of other vectors. This enables for comparison between the different representations. The syntax of sentences is represented by dependency trees, whereas the semantics are captured by Abstract Meaning Representations (AMRs). In addition to the main research question, this thesis aims to answer the following questions:

1. Does the combination of vectors influence the learned representations?
2. When combining vectors, is there a difference between the influence on learned syntax and learned semantics?
3. Can this difference be explained by the underlying architecture of the models?

Answering these questions does not only allow for quantifications of how well a neural representation space corresponds to a structured symbolic representation space, but also enables comparison between various neural representations.

Through Representational Similarity Analysis (RSA) (Kriegeskorte, Mur, & Bandettini (2008), neural representation spaces were quantitatively compared to structured symbolic representation spaces. Moreover, the neural representations seem to capture semantics to a larger extent as opposed to syntax. Additionally, it is shown that encoders might be biased to capture either the syntax or semantics. Combining the vectors which capture different aspects lead to a fuller representation of both syntax and semantics in the combined vectors. These results show a new perspective on the learned representations, which can be used to easily quantify to what extent a neural representation corresponds to other representations.

This thesis is structured as follows. First, a theoretical background will be given of the construction of sentence vectors and their evaluation. Subsequently, the models used in this work will be explained. Next, the experimental setup will be reported, including a description of the data

set and the specific applications of the models to it. Afterwards, the results will be reported, followed by a section which evaluates these results with regard to the research questions. Finally, a concluding summary will be given, along with suggestions for future research.

The code for the analyses in this thesis is available at https://github.com/dennisdegroot/representational-similarity-nn.

2. Background

2.1 Linguistic Vectors

Researchers began to study computational methods for compositionality when it became clear that words could be converted into vectors (Schütze, 1993). Vectors could be constructed by making a matrix of co-occurrences of all words in a corpus and apply dimensionality reduction on it, e.g. matrix factorization techniques (Schütze, 1998; Pennington, Socher, & Manning, 2014). This transformation to vectors enabled for inference through simple linear algebra. One famous example is the analogy "king - man + woman = X", which, using linear algebra on word vectors, results in a vector which is similar to that of "queen" (Mikolov, Yih, & Zweig, 2013a).

Although linear algebra can be used to demonstrate compositionality of word vectors, it cannot be extended to sentences easily. For example, consider the following two sentences "the leaves fell from the tree" and "the tree fell from the leaves". By simply summing the word vectors of the individual words or averaging over them, the two sentences would be represented by the same vector. However, the meaning of both sentences is quite different, and the latter sentence, though being grammatically correct, also makes little sense in English.

One early approach to overcome this issue of semantic compositionality in vector-based models was presented in the work of Mitchel and Lapata (2008), in which they show that models with a multiplicative component outperform additive models. Other approaches assign different representations to different parts of speech. Often nouns are represented by vectors whereas relational words, such as adjectives and verbs, are represented by matrices. Applying the latter to the former results in effective models for constructing vectors representing sentences (Baroni & Zamparelli, 2010; Coecke, Sadrzadeh, & Clark, 2010; Grefenstette & Sadrzadeh, 2011).

Socher, Huval, Manning, and Ng (2012) take a similar approach by assigning both a vector and a matrix to every word in the parse tree of a sentence. The inherent meaning of the word is captured in the vector, whereas the matrix captures the influence of that word on neighbouring

words or phrases. This assignment of vector-matrix representations to all words instead of discriminating between different categories of part of speech ensures greater flexibility. In addition, they are among the first to construct a sentence-based model using a RNN.

2.2 Neural Networks in NLP

While some researches focused on extending word vectors to sentence vectors (e.g. Mitchel & Lapata, 2008), others were optimizing the construction of word vectors by using neural networks (e.g. Collobert & Weston, 2008; Mnih & Hinton, 2009; Turian, Ratinov, & Bengio, 2010; Mikolov, Chen, Corrado, & Dean, 2013b). Elman (1990) introduced RNNs in 1990 to model the temporal dimension. By changing the output-to-memory recurrent connections in the architecture of Jordan (1986) to hidden-to-memory recurrent connections, he enabled his network to represent dynamic systems. He later showed that a RNN can encode lexical categories, relevant grammatical relations and hierarchical constituent structure (Elman, 1991).

The early models for constructing word vectors using neural networks, e.g. Collobert and Weston (2008), assigned an initial vector to each given word, which was modified depending on other words in a context, resulting in a vector which is used to predict other words in the context (Le & Mikolov, 2014). After single words could be encoded using such models, researchers tried to capture larger linguistic concepts. For example, Socher, Pennington, Huang, Ng, and Manning (2011) used recursive autoencoders to predict the sentiment of sentences. Based on this model, Socher et al. (2012) were able to construct their earlier discussed matrix-vector RNN which could learn compositional vector representations for phrases and sentences.

A novel adaptation of a RNN architecture called Long Short-Term Memory (LSTM), provided by Hochreiter and Schmidhuber (1997), had proven to be able to process complex sequences with long-range structure (Graves, 2013). By modifying the units in a neural network with a memory cell and four gating units, the information flow inside the unit could be controlled. This allowed neural networks to shift attention and memory through different parts of their input, by focussing on predicting one data point at a time.

Cho et al. (2014) proposed a RNN Encoder-Decoder, a neural network model focussed on statistical machine translation, with two novel ideas. First, the model consists of two RNNs: one which is able to encode a sentence into a fixed-length vector representation, and one which decodes such a vector representation into another sequence of symbols. In addition to this novel model

architecture, they also proposed a new type of hidden unit. Inspired by the LSTM unit, they proposed the Gated Recurrent Unit (GRU). The underlying idea is that an update gate selects whether the hidden state should be updated, and a reset gate decides whether the previous hidden state is ignored. This enables control of information transfer between hidden states, similar to LSTM networks. However, GRU is simpler to compute and implement (Cho et al., 2014).

Inspired by encoder-decoder models used for neural machine translation, such as the model of Cho et al. (2014), a seminal approach called Skip-Thoughts was introduced by Kiros et al. (2015). Abstracting the skip-gram model of Mikolov et al. (2013b), which learns word vectors to predict its surrounding context, to a sentence level, the skip-thought model is able to encode a sentence to predict its neighbouring sentences. The encoder consists of a RNN with GRU, which processes the sentence. The decoder is trained to predict the preceding and following sentence, based on the final hidden state of the encoder. The skip-thought model will also be used in this thesis and it will be explained in more detail in Section 3.1.

2.3 Evaluation of Linguistic Representations

The rise of RNNs in various NLP domains, such as parsing (Vinyals et al., 2015) and machine translation (Bahdanau et al, 2015), built upon the fact that variable-length linguistic expressions could be represented by encoding them into a fixed-size low-dimensional vector (Kádár et al., 2017). The nature of the encoding of RNNs is often complex and non-linear, making it difficult to interpret their mechanisms as well as being able to control and account for them. Uncovering the underlying structure of the learned representations, as well as investigating how abstract linguistic knowledge can be learned are therefore increasingly being studied. This can be done via two approaches.

The first approach focuses on examining hidden activation patterns of RNNs. For example, Kádár et al. (2017) estimate the salience of a word by comparing the representation of a sentence omitting that word to the representation of the original sentence. By doing so, they also found that models can learn various kinds of linguistic features besides lexical cues, e.g. paying more attention to syntactic structures instead of the linear order of words in a sentence. Similarly, Li, Monroe and Jurafsky (2016) analysed the impact of individual input tokens, hidden units and word embedding dimensions by removing them from the representations and evaluated how the model was affected by it. They found that some models focus more on specific dimensions of word vectors, and that

some dimensions are important for multiple feature classifications tasks, such as detection of prefixes and suffixes. By comparing different representations learned by models and qualitatively comparing visualisations of hidden unit activations, the intrinsic workings of the network are examined (Belinkov, Durrani, Dalvi, Sajjad, & Glass, 2017).

Instead of comparing different model representations, the hidden states of neural networks can also be used to assess learned linguistic features. For example, Broere (2018) evaluates the syntactic properties of the skip-thought model, which is also used in this thesis. A logistic regression trained on the hidden states of the model proved capable of classifying different grammatical categories. Not only does this shed light on the learned syntax by the skip-thoughts model, but it also shows that the hidden states of neural networks can be used to examine the learned syntax in different ways.

On the other hand, the second approach inspects the learned representations similar to studying human language processing by analysing the behaviour on targeted sentences to evaluate specific aspects of the learned representations (Linzen et al., 2016). This can be done for example by an agreement prediction task (Bock & Miller, 1991), as used by Linzen et al. (2016). An example of such a task is to finish the sentence "The key to the cabinets…", which should grammatically be followed by "was" instead of "were". By reviewing performance on comparable tasks, representations are extrinsically evaluated on learned syntactic or semantic knowledge.

Both approaches have been proven useful for relating neural representations to symbolic structures from linguistic theory, such as syntax and syntactic trees, as well as lexical categories and affixes. Often representations are trained and evaluated on specific structures using a variety of different downstream tasks. Although this leads to a deeper understanding of the learned knowledge of that particular structure or performance on that task, comparing them on a more general level can be difficult.

To overcome this problem, multiple toolkits have been introduced for a more centralized way for evaluating representations, such as SentEval (Conneau & Kiela, 2018) and GLUE (Wang et al., 2018). These toolkits, as well as similar approaches (Conneau, Kruszewski, Lample, Brault, & Baroni, 2018; McCann, Keskar, Xiong, & Socher, 2018), use a combination of probing tasks to assess the extrinsic performance of neural representations. If models are evaluated along the same lines by assessing them using identical tasks, comparisons between them can easier be made.

Although these frameworks enabled evaluations of representations in a more systematic way, they still compare performance on specific downstream tasks to assess which particular syntactic rules are learned by the models. Models need to be trained on established training data prior to evaluation on multiple assignments. Not only can this be a laborious process, it still leaves questions about the learned representations unanswered. For example, a model can perform well on several agreement prediction tasks, but to what extent does it capture the overall syntax of a language? Although being able to correctly predict other words or sentences, how well can a model learn the overall semantics? And how are these symbolic representation spaces, such as semantics and syntax, related? In other words: to what extent does a model capture semantics as opposed to syntax? This thesis aims to answer these questions by proposing a simple and more global approach to quantify how well a neural representation space corresponds to a structured symbolic representation space.

The approach proposed in this study applies RSA to a number of neural and structured symbolic representations of sentences. RSA, as introduced by Kriegeskorte et al. (2008) will be explained in the Section 3. Methods of constructing the various representations will also be explained in that section. A description of the used data set and the experimental procedure will be given in Section 4.

3. Methods

In this section, the approaches to constructing the various representations will be explained, as well as the analysis used to compare these representations. First, a description will be given of the neural representation of sentences. Subsequently, the structured symbolic representations will be described. Finally, the technique used for comparing these representations, RSA, will be explained.

3.1 Neural Representations

In this thesis, neural representations of sentences are constructed by encoding sentences into vectors using the skip-thoughts model provided by Kiros et al. (2015). This model was built and trained to, given a sentence, predict the preceding and following sentence. The model consists of an encoder, which maps a sentence into a vector, and a decoder, which translates this vector back to a sentence. An illustration of the model can be found in Figure 1. Since the focus of the current work is on comparing neural representations, only the encoder is considered.

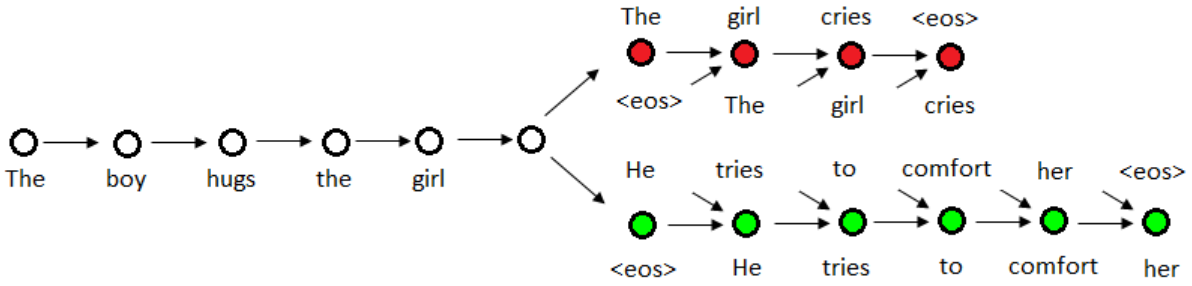


Figure 1. The skip-thoughts model. Given a tuple (s_{i-1}, s_i, s_{i+1}) of consecutive sentences, with $s_i$the i-th sentence of a dataset, which is encoded and used to reconstruct the preceding sentence s_{i-1} and next sentence s_{i+1}. In this example, the input is the sentence triplet "The girl cries.", "The boy hugs the girl.", "He tries to comfort her." The unattached arrows are connected to the encoder output and the colours indicate which components share parameters. <eos> is the end of sentence token.

The encoder, which is a RNN with GRU activations, works as follows, following Kiros et al. (2015). Let $w_i^1, \ldots, w_i^N$ be the words in sentence s_i where N is the number of words in the sentence and let $\mathbf{x}_i^t$ be the word embedding of w_i^t. A hidden state $\mathbf{h}_i^t$ is produced by the encoder at each time step, which can be interpreted as the representation of the sequence $w_i^1, \ldots, w_i^t$. Therefore, $\mathbf{h}_i^N$ represents the full sentence s_i. Sentences are encoded by iterating over the following sequence of equations. The subscript i is dropped for readability.

$$\mathbf{r}^t = \sigma(\mathbf{W}_r\mathbf{x}^t + \mathbf{U}_r\mathbf{h}^{t-1}) \quad (1)$$

$$\mathbf{z}^t = \sigma(\mathbf{W}_z\mathbf{x}^t + \mathbf{U}_z\mathbf{h}^{t-1}) \quad (2)$$

$$\overline{\mathbf{h}}^t = \tanh(\mathbf{W}\mathbf{x}^t + \mathbf{U}(\mathbf{r}^t \odot \mathbf{h}^{t-1})) \quad (3)$$

$$\mathbf{h}^t = (1 - \mathbf{z}^t) \odot \mathbf{h}^{t-1} + \mathbf{z}^t \odot \overline{\mathbf{h}}^t \quad (4)$$

where $\overline{\mathbf{h}}^t$ is the proposed state update at time t, $\mathbf{z}^t$ is the update gate, $\mathbf{r}^t$ is the reset gate, and $\mathbf{W}$ and $\mathbf{U}$ are weight matrices which are learned. $\odot$ denotes a component-wise product. Both update gates take values between zero and one.

The skip-thoughts model consists of two separately trained encoding models. The first one, uni-skip, is a unidirectional encoder with 2400 dimensions. The other is a bidirectional model, bi-

skip, which is made up by a forward and a backward encoder of 1200 dimensions each. The forward encoding component of this model, bi-skip forward, is given the sentence in correct order, whereas the backward encoder, bi-skip backward, handles the sentence in reverse order. The two encoded vectors of the uni-skip and bi-skip models can be combined to form a concatenated vector of 4800 dimensions, which will be referred to as combi-skip.

3.2 Structured Symbolic Representations

In this study, two structured symbolic representations are constructed. One focussing on the syntax of a sentence, whereas the other represents the semantics.

3.2.1 Syntax

The syntax of sentences is represented by a dependency tree, a syntactic representation that denotes the grammatical relations between words (Moschitti, 2006). Figure 2 shows the dependency tree of the sentence "The boy hugs the girl". Parsing a sentence to construct such a dependency tree is called dependency parsing.

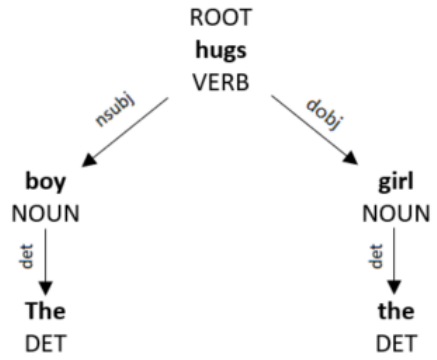


Figure 2. Dependency tree of the sentence "The boy hugs the girl". The words in bold are the words from the original sentence and below them are the corresponding part of speech tags. The edges are tagged with the syntactic dependency relation.

The parser used in this study to construct dependency trees is the spaCy parser (Honnibal & Montani, 2018). This parser is an updated version of the parser of Honnibal and Johnson (2015).

Since the current architecture has not been published yet, we refer to the website of spaCy for a description of the parser and its code[1].

3.2.2 Semantics

To represent the semantics of a sentence, Abstract Meaning Representation (AMR) is used. This semantic representation language, as presented by Banarescu et al. (2013), is constructed to abstract away from syntactic idiosyncrasies and attempts to assign the same AMR to sentences with the same basic meaning. The same AMR is assigned to the sentences “he described her as a genius” and “she was a genius, according to his description”. Therefore, AMR is used to represent sentences in such a way that the underlying meaning of a sentence, its semantics, is captured well.

AMRs are written as rooted, directed, edge-labeled, leaf-labeled graphs. This traditional format is similar to simple forms of feature structures (Shieber et al., 1986), conjunctions of logical triples, directed graphs and PENMAN inputs (Matthiessen & Bateman, 1991). Figure 3 illustrates some of these views for the sentence “The boy hugs the girl”.

[1] https://spacy.io/

LOGIC format:

∃ h, b, g:
instance(a, hug-01) ^
instance(b, boy) ^
instance(c, girl) ^
ARG0(a, b) ^
ARG1(a, c)

AMR format:

(h / hug-01
:arg0 (b / boy)
:arg1 (g / girl))

GRAPH format:

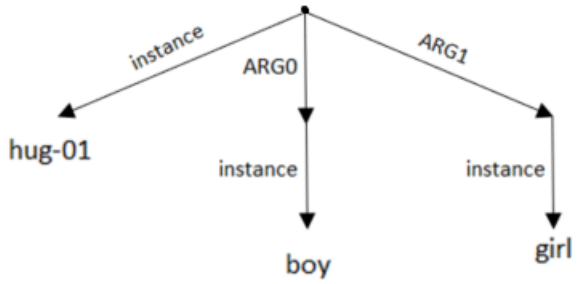


Figure 3. Equivalent formats for representing the meaning of "The boy hugs the girl".

3.3 Representational Similarity Analysis

The technique used in this study to compare the different representations is called Representational Similarity Analysis (RSA). In this analysis, introduced by Kriegeskorte et al. (2008), a Representational Dissimilarity Matrix (RDM) is constructed for each model. The RDM contains cells for all pairs of representations of the entries of a data set, where the value of the cell reflects the dissimilarity between those two representations. Consequently, a RDM is symmetrical about a diagonal of zeros. An example of a RDM is given in Figure 4.

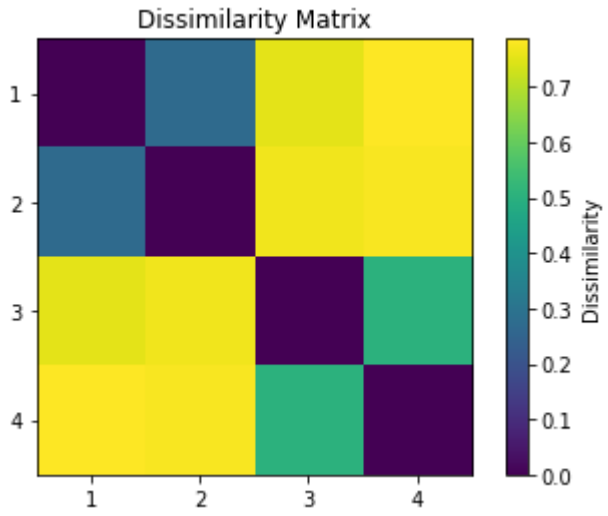


Figure 4. Example of a RDM for a data set of four sentences: 1: “What a nice man”, 2: “What a beautiful woman”, 3: "The dog barks", 4:"The cat eats". Each sentence is represented by a vector using the skip-thoughts model. Each cell reflects the dissimilarity between the two sentence representations, which is computed using the cosine distance.

Having constructed RDMs for the various models, these representations can be quantitatively compared by computing the correlation between the RDMs. Since the RDMs are symmetrical about a diagonal of zeros, the correlation will be computed over the upper triangular of the matrices. How these RDMs are constructed in this study specifically, will be explained in the next section.

4. Experimental Setup

In this section, the experimental procedure will be explained. First, a description of the data set will be given. Subsequently, the construction of the different representations and their RDMs will be explained. All coding was done in Python.

4.1 Data Set

The data used in this work consists of The Little Prince Corpus[2]. This data set contains all sentences of the novel The Little Prince by Antoine de Saint-Exupéry, which was published in 1943. This data set was chosen because it provides AMR annotations[3] for all sentences. The corpus consists of 1562 sentences, which are already split into a training set of 1274 sentences, a development set of 145 sentences, and 143 test sentences. Since no models are trained in this thesis, a development or test set is not needed. Therefore, only the training set was used.

The data set consists of sentences in English, their Chinese translations, their AMR and some metadata, including the annotators id and save date. The data was pre-processed so that only the English sentences and the AMRs remained. Chapter headings, such as "Chapter 7", were excluded, since they are not full sentences that contribute to the story of the novel, after which the total amount of sentences was reduced to 1253.

4.2 Representations

4.2.1 Neural representations

As described before, the neural representations used in this study are skip-thought vectors (Kiros et al., 2015). The code for the encoder used in the paper as well as the encoder vocabulary are freely available[4]. After having installed the skip-thoughts model, sentences can easily be encoded into a vector with 4800 dimensions. Although other models have been introduced more recently (e.g. the quick thoughts model of Logeswaran and Lee, 2018), skip-thoughts was chosen because of time limitations, since it is easy to use and produces a combination of vectors. This enables comparison between vectors after having only one model installed.

Since the produced vector is a combination of other vectors, the subvectors can be easily extracted, allowing for comparisons between these vectors. The complete vector of 4800 will be referred to as combi-skip vector. This vector is made up of two vectors, each of 2400 dimensions. The first, a unidirectional encoded vector, will be referred to as the uni-skip vector. The second, a bidirectional encoded vector, will be referred to as the bi-skip vector. By selecting the first or last 2400 dimensions of the combi-skip vector, these vectors can be extracted. The bi-skip vector is

[2] https://amr.isi.edu/download.html

[3] The annotations, provided by the AMR Bank, are manually constructed by human annotators at the Linguistic Data Consortium, SDL, the University of Colorado's Center for Computational Language and Education Research, and the University of Southern California's Information Science Institute and Computational Linguistics at USC.

[4] https://github.com/ryankiros/skip-thoughts

also a combination of two vectors of 1200 dimensions each. The first one is constructed by an encoder which is given the sentence in correct order, whereas the other is given the sentence in reverse. The produces vectors will be referred to as the bi-skip forward vector and bi-skip backward vector respectively. In summary, 5 vectors with the following number of dimensions will be extracted: the combi-skip vector (4800), the uni-skip vector (2400), the bi-skip vector (2400), the bi-skip forward vector (1200), and the bi-skip backward vector (1200).

4.2.2 Syntactic representations

Dependency trees are the syntactic representations used in this study. The dependency parser used is spaCy, which code is freely available[5]. This parser is pre-trained and can easily be used. After having installed the model and loading the English language model, each sentence was encoded individually and the results were stored in a list.

To evaluate the dependency trees, a tree kernel was used, which will be explained in Section 4.3. This kernel required the input to be in CoNLL-U format[6]. Therefore, some features were extracted from the encoded dependency parse. For each word in the sentence, its index, text, lemma, coarse-grained Part-Of-Speech (POS) tag, fine-grained POS tag, dependency, head, and the index of the head were extracted. The extracted features were rewritten in CoNLL-U format, after which these rewritten representations were stored in a list. Although they were rewritten to another format, they still captured the dependency parse and implicitly the dependency tree of the sentence.

4.2.3 Semantic representations

The semantic representations used are AMRs. Since they were provided in the data set, these did not need to be constructed.

4.3 RSA

[5] https://spacy.io/
[6] http://universaldependencies.org/format.html

In order to apply RSA to the representations, first the RDMs for the various representations had to be constructed. These RDMs were constructed of pairwise dissimilarity measurements between the all sentences of the corpus. The resulting matrix has a shape of (1253, 1253) and is symmetrical about a diagonal of zeroes. The dissimilarity measurements differ for the various representations and will now be explained.

The neural representations are made up of vectors. Therefore, their dissimilarity was measured in cosine distance, i.e. $1 - \frac{u \cdot v}{\|u\|_2 \|v\|_2}$.

The syntactic representations were constructed in the form of dependency trees. These were evaluated using tree kernels as described in Moschitti (2006). The script for this tree kernel can be found on GitHub[7]. This kernel was adapted for dependency trees. The kernel evaluates trees in terms of their substructures. The substructures can be characterized in two ways: the subtrees (STs) and the subset trees (SSTs). A ST is any node of a tree and all its descendants, whereas a SST is a more general structure in which leaves can be associated with non-terminal symbols. An illustration of a tree with some of its STs and SSTs can be found in Figure 5 and Figure 6 respectively.

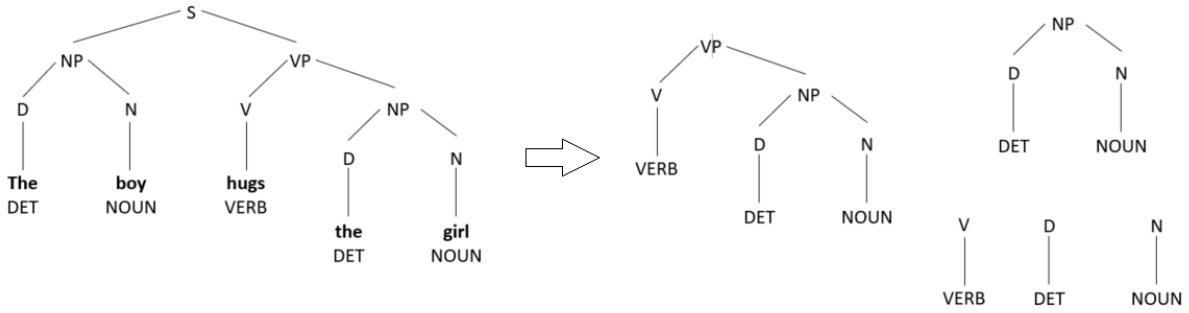
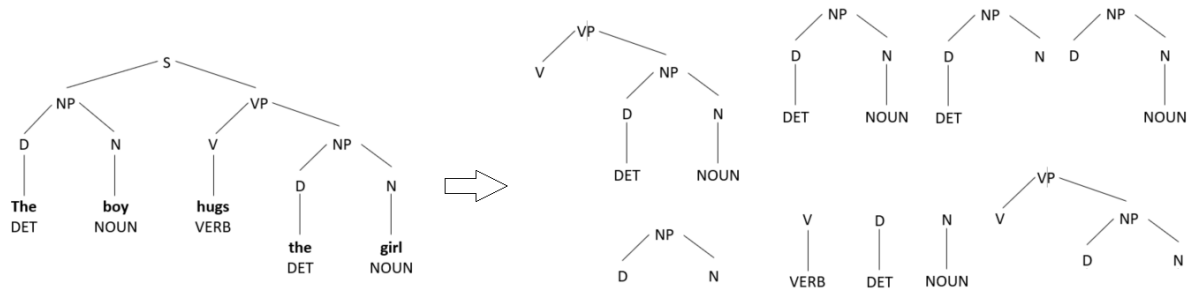


Figure 5. A tree with some of its subtrees (STs).



[7] https://github.com/fkunneman/DiscoSumo/tree/master/naacl/models

Figure 6. A tree with some of its subtrees (SSTs).

The kernel constructs a feature space and detects if a tree subpart, common to both trees, belongs in that space. Tree kernels compute the number of common substructures between two trees. Although SST kernels provide a much higher accuracy than ST kernels on classification of predicate argument structures (Moschitti, 2006), both kernels are explored.

The tree kernel, which is a convolutional kernel, works as follows, following Moschitti (2006). Let $F = \{f_1, f_2, \dots, f_{|F|}\}$ be a set of tree fragments of any substructure and let indicator function $I_i(n)$ be 1 if the target f_1 is rooted at node n and 0 otherwise. The kernel can be defined as:

$$K(T_1, T_2) = \sum_{n_1 \in N_{T_1}} \sum_{n_2 \in N_{T_2}} \Delta(n_1, n_2) \tag{5}$$

where N_{T_1} and N_{T_2} are the sets of nodes in T_1 and T_2 respectively, and $\Delta(n_1, n_2) = \sum_{i=1}^{|F|} I_i(n_1) I_i(n_2)$, which is the number of common fragments rooted at nodes n_1 and n_2. It can be computed as follows:

1. if the productions at node n_1 and n_2 are different then $\Delta(n_1, n_2) = 0$;
2. if the productions at node n_1 and n_2 are identical, and both have only leaf children, i.e. they are pre-terminal symbols, then $\Delta(n_1, n_2) = 1$;
3. if the productions at node n_1 and n_2 are identical, and they are not pre-terminals, then

$$\Delta(n_1, n_2) = \prod_{j=1}^{nc(n_1)} (\sigma + \Delta(c_{n_1}^j, c_{n_2}^j)) \tag{6}$$

where $\sigma \in \{0,1\}$, $nc(n_1)$ is the number of children of n_1 and c_n^j is the j-th child of node n. Since the productions are identical, $nc(n_1) = nc(n_2)$.

If $\sigma = 0$, then $\Delta(n_1, n_2)$ is equal to 1 only if all the productions associated with the children are identical, i.e. $\forall j\ \Delta(c_{n_1}^j, c_{n_2}^j) = 1$. Recursive application of this property shows that n_1 and n_2 have identical subtrees. Therefore, the first equation evaluates the ST kernel. The SST kernel can be computed when $\sigma = 1$, so that the number of SSTs common to both n_1 and n_2 are evaluated (Collins and Duffy, 2002).

A similarity score between 0 and 1 can be computed by applying normalization in the kernel space, i.e. $K'(T_1, T_2) = \frac{K(T_1,T_2)}{\sqrt{K(T_1,T_1) \times K(T_2,T_2)}}$. The dissimilarity score was computed by $1 - K'(T_1, T_2)$.

The semantic representations in the form of AMRs were evaluated using smatch, a metric which calculates the degree of overlap between two semantic feature structures (Cai & Knight, 2013). Consider two sentences, "the boy hugs the girl" and "the boy wants the football", which AMR graphs are illustrated as conjunctions of triples in Figure 7. The metric measures the amount of propositional overlap between the two sentences. Since variable names are not necessarily shared between two AMRs, overlap can be computed on different variable mappings. Therefore, the smatch score is defined as the maximum F-score obtainable via a one-to-one matching of variables between two AMRs. An example is given in Table 1.

```
instance(a, hug-01)  ^        instance(x, want-01)  ^
instance(b, boy)  ^           instance(y, boy)  ^
instance(c, girl)  ^          instance(z, football)  ^
ARG0(a, b)  ^                 ARG0(x, y)  ^
ARG1(a, c)                    ARG1(x, z)
```

Figure 7. The AMRs illustrated as conjunctions of triples for the sentences "the boy hugs the girl" (left) and "the boy wants the football" (right).

Table 1

Computation of Smatch Score for Two Sentences

Combination	M	P	R	F
x = a, y = b, z = c	3	3/5	3/5	0.6
x = a, y = c, z = b	0	0/5	0/5	0.0
x = b, y = a, z = c	0	0/5	0/5	0.0
x = b, y = c, z = a	0	0/5	0/5	0.0
x = c, y = a, z = b	0	0/5	0/5	0.0
x = c, y = b, z = a	1	1/5	1/5	0.2
smatch score				0.6

Note. The precision, recall and F-score for different mappings of the two sentences "the boy hugs the girl" and "the boy wants the football", where M is the number of propositional triples that agree given a variable mapping, P is the precision of the second AMR against the first, R is the recall, and F is the F-score. The smatch score is the maximum of the F-scores.

The smatch evaluation script is freely available[8]. The script can evaluate the semantic match between all triples or specific triples. By adding hyperparameters, the script can focus on specific triples, such as instances, relations, or attributes, instead of calculating the score over all triples in the AMR. An example of instance triples can be seen in Figure 7, where the upper 3 triples in both AMRs are instance triples. The bottom two illustrate relation triples. An example of an attribute is negation. For example, the sentence "the boy does not hug the girl" would have an additional attribute for the triple ARG0(a, b) compared to the first sentence in Figure 7. Although all hyperparameters were explored, specific focus was on the evaluation over all triples, since this captures the complete underlying meaning of the sentences.

Besides the computed F-score, the smatch script can also return the corresponding precision and recall. However, since these scores depend on the order of the presented AMR, so that the precision of AMR_i against AMR_j is equal to the recall of AMR_j against AMR_i, only F-score was selected. The dissimilarity between a pair of AMRs was therefore computed by $1 - \text{smatch score}$.

[8] https://github.com/snowblink14/smatch

Having constructed the RDMs for the different representations, the Pearson correlation coefficient was computed between two RDMs on the upper triangulars of the matrices, excluding the diagonal.

5. Results

The computed RDMs for the neural representations and structured symbolic representations can be found in Appendix A and Appendix B respectively. Table 2 shows the RSA scores between the neural representations and the structured symbolic representations. Comparing the RSA scores between the neural representations and the structured symbolic representations show that all neural representations are more correlated to all semantic representations than to the syntactic representations. Moreover, the difference between the correlations of the vectors with the different structured symbolic representations is quite large. The syntactic correlations do not exceed an absolute value of .021, with one exception of .040, while the semantic correlations do not drop below a correlation coefficient of .055. Note that the syntactic evaluations using the ST kernel show positive correlations with bi-skip and bi-skip forward, as well as negative correlations with the other representations.

Focussing on the syntactic representations, comparison between the RSA scores of the neural representations and the syntactic representation evaluated using the SST kernel shows that the combi-skip (r = .017, 95% CI [.014, .019]) and the uni-skip representations (r = .017, 95% CI [.015, .019]) are more correlated to the syntactic representation than the bi-skip representation (r = .015, 95% CI [.013, .017]), which on its turn has a higher correlation than its sub-models (bi-skip forward: r = .006, 95% CI [.003, .008]; bi-skip backward: r = .010, 95% CI [.008, .013]). Regarding the RSA scores of the neural representations and the syntactic representation evaluated using the SST kernel, the uni-skip representation has the largest absolute correlation coefficient (r = -.040, 95% CI [-.042, -.038]), followed by bi-skip backward (r = -.021, 95% CI [-.023, -.019]) and bi-skip forward (r = .021, 95% CI [.018, .023]). The combi-skip follows closely (r = -.018, 95% CI [-.020, -.016]), though the bi-skip representation has a substantially lower absolute correlation (r = .007, 95% CI [.004, .009]).

Table 2

Pearson Correlation Coefficients between RDMs of Neural Representations and Structured Symbolic Representations

Structured Symbolic Representations	Neural Representations				
	Combi-skip	Uni-skip	Bi-skip	Bi-skip f	Bi-skip b
Dep SST	.017	.017	.015	.006	.010
Dep ST	-.018	-.040	.007	.021	-.021
AMR all	.134	.087	.175	.143	.153
AMR instance	.093	.064	.117	.072	.113
AMR relation	.093	.055	.126	.115	.104
AMR attribute	.077	.055	.096	.076	.087

Note. Bi-skip f and Bi-skip b refer to the bi-skip forward vector and bi-skip backward vector respectively. Dep SST and Dep ST refer to syntactic representations evaluated using the subset tree kernel and subtree kernel respectively. The different AMRs are the semantic representations in the form of AMRs focussing on all triples, instance triples, relation triples, and attribute triples respectively.

Comparison between the RSA scores of the neural representations and the semantic representation evaluated using AMRs focussing on all triples shows that all bi-skip representations have a higher correlation with this semantic representation (bi-skip: r = .175, 95% CI [.173, .177]; bi-skip forward: r = .143, 95% CI [.141, .145]; bi-skip backward: r = .153, 95% CI [.151, .155]) than the combi-skip (r = .134, 95% CI [.132, .137]) and uni-skip representations (r = .087, 95% CI [.085, .089]). The combined bi-skip model has a higher correlation than both of its submodels, bi-skip forward and bi-skip backward. Uni-skip scores considerably lower than all other representations. Comparing the RSA scores of the neural representations and the other semantic representations, a similar pattern can be observed, although the correlations are smaller.

5.1 Dissimilarity Score Distributions

Further investigation of the distribution of the dissimilarity scores, which can be found in Figure 8, show that both structured symbolic representations are highly skewed towards higher dissimilarity scores as opposed to the neural representations. Moreover, all neural representations seem to reach a global maximum, followed by a substantial decline, after which it reaches a new local maximum. Almost all neural representations follow this pattern, except bi-skip forward, which second maximum is slightly higher than the first. The differences in dissimilarity distributions between the structured symbolic representations and the neural representations can also be seen in Table 3, which illustrates the mean and standard deviation of the dissimilarities for the representations. The mean scores for the structured symbolic representations are much higher with a lower standard deviation ($M_{\text{AMR}} = 0.857$, $SD_{\text{AMR}} = 0.096$; $M_{\text{DEP}} = 0.966$, $SD_{\text{DEP}} = 0.052$), as opposed to the neural representations, which have lower mean scores and higher standard deviations ($M_{\text{Combi-skip}} = 0.458$, $SD_{\text{Combi-skip}} = 0.162$; $M_{\text{Uni-skip}} = 0.399$, $SD_{\text{Uni-skip}} = 0.171$; $M_{\text{Bi-skip}} = 0.517$, $SD_{\text{Bi-skip}} = 0.164$; $M_{\text{Bi-skip f}} = 0.496$, $SD_{\text{Bi-skip f}} = 0.277$; $M_{\text{Bi-skip b}} = 0.512$, $SD_{\text{Bi-skip b}} = 0.160$).

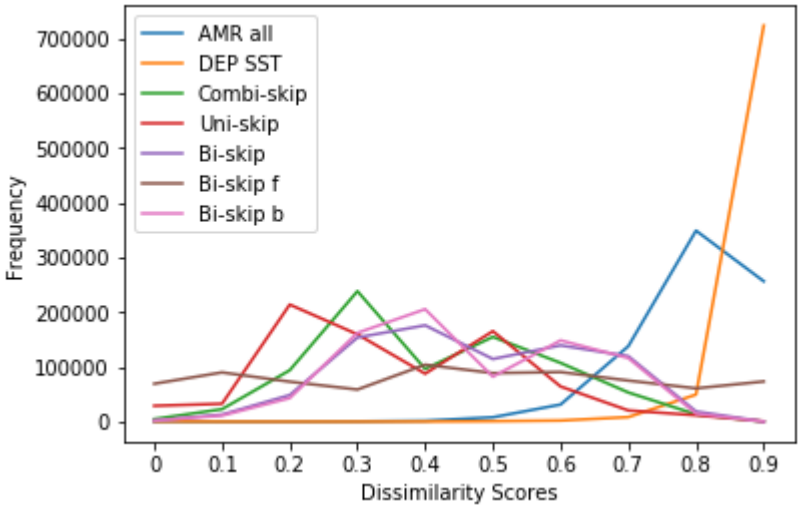


Figure 8. Distribution of dissimilarity scores over the upper triangular of the RDM for different representations. The numbers on the x-axis indicate the lower bound of the bin range, whereas the y-axis indicates the number of dissimilarity scores in that bin. The dissimilarity scores are binned into 10 bins with a width of 0.1, so that the frequency for a given value on the x-axis reflects the number of dissimilarity scores from that value up to the next value for that representation. AMR

all and DEP SST refer to the semantic representation using AMRs focussing on all triples and the syntactic representation evaluated using the subset tree kernel respectively. The other lines represent the different neural representations of the sentences.

Table 3

Mean and Standard Deviation of the Dissimilarity Scores for Representations

Representation	Mean	Std. Dev.
AMR	0.857	0.096
DEP	0.966	0.052
Combi-skip	0.458	0.162
Uni-skip	0.399	0.171
Bi-skip	0.517	0.164
Bi-skip f	0.496	0.277
Bi-skip b	0.512	0.160

Note. AMR and DEP refer to the semantic representation using AMRs focussing on all triples and the syntactic representation evaluated using the subset tree kernel respectively. The bottom five rows represent the different neural representations of the sentences.

Table 4 and Table 5 show several sentences of the data set and the dissimilarity scores of some combinations. It shows that the combination of sentences (7, 23) has the same syntactic structure and has a semantic dissimilarity of 0.400. The dissimilarities of the neural representations are also low. Another combination which yields low dissimilarities for the neural representations is (361, 362), which also has a rather low semantic dissimilarity of 0.429, yet has a higher syntactical dissimilarity of 0.792. This is still lower than the syntactic dissimilarity of the other combinations.

Furthermore, both combinations (7, 8) and (361, 362) are consecutive sentences, of which the first combination is much more dissimilar than the second. This holds for all representations. Surprisingly, when both sentences of this first combination are compared to one of the second combination, i.e. (7, 361) and (8, 361), their dissimilarity scores across all representations are very similar.

Table 4

Sentences of the Little Prince Corpus with their Sentence ID

ID	Sentence
7	I did not know .
8	At that moment I was very busy trying to unscrew a bolt that had got stuck in my engine .
23	I did not answer .
361	FIfteen and seven make twenty - two .
362	Twenty - two and six make twenty - eight .

Note. The sentence ID is the ID after the removal of chapter headings, as discussed in Section 4.1. The sentences are the original sentences from the dataset, including the space around interpunction and the two capital letters at the beginning of sentence 361.

Table 5

Dissimilarity Scores between Combinations of Sentences

s1	s2	AMR	DEP	Combi-skip	Uni-skip	Bi-skip	Bi-skip f	Bi-skip b
7	8	0.852	0.989	0.298	0.260	0.336	0.192	0.352
7	23	0.400	0.000	0.153	0.159	0.148	0.010	0.161
7	361	1.000	0.945	0.355	0.310	0.400	0.169	0.425
8	361	0.931	0.997	0.362	0.310	0.414	0.323	0.426
361	362	0.429	0.792	0.113	0.100	0.127	0.040	0.139

Note. s1 and s2 represent the IDs of the first and second sentence of the combination. AMR and DEP refer to the dissimilarity scores of the semantic representation using AMRs focussing on all triples and the syntactic representation evaluated using the subset tree kernel respectively. The last five columns represent the dissimilarity between the different neural representations of the sentences.

6. Discussion

In this study, a simple approach was explored to quantify how well a neural representation corresponds to a structured symbolic representation. Through RSA neural representations were compared to representations capturing syntax and semantics of sentences, in the form of dependency trees and AMRs respectively. The used neural representations were skip-thought vectors and its subvectors (Kiros et al., 2015). This analysis was conducted to answer the following research question: "To what extent captures a neural network representation the syntax of sentences as opposed to the semantics?"

Regarding the RSA scores between the neural representations and the syntactic representations, the correlations with the ST kernel show a surprising result. They show both positive and negative correlations. In addition, SST kernels provide a better accuracy on classification of predicate argument structures (Moschitti, 2006). Therefore, the rest of this thesis will focus on the correlations between the neural representations and the SST kernel evaluations for the syntactic representations. For simplicity, in the rest of this thesis the syntactic representations will refer to the representations constructed by the SST kernel.

On the other hand, the correlations for the various neural representations show a similar pattern between all semantic representations. The only exceptions are the combi-skip vector and bi-skip forward vector, which outperform each other on different semantic representations. Moreover, the evaluation focussing on all triples captures the meaning of the complete sentence, as opposed to focussing on specific types of represented meanings. In addition, this evaluation yields the highest correlation with all neural representations, indicating that the vectors seem to capture the complete meaning of sentences to a larger extent as opposed to the semantics captured by specific triples. Therefore, despite the fact that there is small variation in the correlation patterns between the neural representations and the semantic representations, the rest of this thesis will focus on the correlations between the neural representations and the evaluation focussing on all triples. For simplicity, in the rest of this thesis the semantic representations will refer to the representations constructed by evaluation on all triples.

As discussed in the previous section, the combi-skip and the uni-skip models have a higher correlation with the syntactic representation than the bi-skip models. In addition, the combined bi-skip model has a higher correlation than its two submodels, bi-skip forward and bi-skip backward. This might indicate that combined models capture the syntax as good as or better than their

submodels. One possible explanation might be that the submodels capture different aspects of the syntax and therefore their combination captures a fuller representation of the syntax.

Further inspection of the correlations of the neural representations with the semantic representation shows something different. The bi-skip model, as well as its forward and backward encoders, have a higher correlation with the semantic representation than the uni-skip and combi-skip model. The uni-skip vector correlates considerably less with the semantic representation than all other representations. Since half of the combi-skip vector consists of this vector, it might be negatively impacted by its low correlation with the semantic representation. The combination of the forward and backward vector in the bi-skip vector suggest combining vectors lead to a better representation of the semantics, though this does not hold for the combi-skip vector. Although its lower correlation can be explained by the low correlation of the uni-skip vector, further research is needed to verify if combining vectors increase the captured semantics.

When comparing the correlations between the syntactic and semantic representations, it can be seen that the neural representations are more correlated to semantics as opposed to syntax, thereby answering the main research question. This implies that the meaning of the sentence is captured to a larger extent than the underlying grammar. This might be a reflection of modern English, as well as other languages, in which the specific words contribute more towards a sentence representation of a concept than its syntax. For example, the two sentences “she was a genius, according to his description” and “he described her as a genius” can be used both to represent the same concept, though be it in from a different perspective. Considering a third sentence “he was a dictator, according to his actions”, which is similar to the first sentence in terms of syntax but is different in terms of semantics, represents a very different concept than the first two sentences. This illustrates that the meaning of the words in a sentence contribute to a larger extent to the representation of a concept than the used grammar. This result also holds for neural representations, which capture the semantics to a larger extent as opposed to the syntax, thereby answering our research question.

However, it must be noted that the correlations with the syntactic representation are quite low. Moreover, the RDMs of the neural representation and the syntactic representation have a low, negative correlation, r = -.100. In addition to the explanation from the preceding paragraph, the lower scores for the syntactic representation might also be caused by two limitations of this study. First, it might be affected by the different ways of computing the dissimilarity between the AMRs

and the dependency trees. Computing the F-score over the syntactic dependency triples might allow for a better comparison. In addition, a third tree kernel focussing on another substructure, partial trees, provides a slightly higher accuracy on dependency trees and might improve the representation (Moschitti, 2006, September). This kernel was not explored because the code for this kernel was not available and could not be constructed due to time limitations. Second, the AMRs provided were constructed and checked by human annotators, whereas the dependency trees were constructed automatically. Although the encoder was chosen because it offered an easy way of constructing syntactic representations given the limited time for this study and because its high accuracy (Honnibal & Johnson, 2015), it still might have a negative influence on the results. Future studies might also consider humanly constructed dependency trees, thereby reducing the chance of incorrect syntactic representations. These alternative explanations might indicate that the results are inconclusive, thereby impacting the reliability and generalizability of this conclusion and other conclusions in this thesis based on this result.

Additionally, the distributions of the structured symbolic representations are very skewed as opposed to those of the neural representations. The structured symbolic representations have a higher mean and lower standard deviation in comparison with the neural representations, yet the dissimilarity scores for all representations are in the same range between 0 and 1. Therefore, the lower dissimilarities for structured symbolic representations seem to be outliers, which might have an influence on the computed RSA score.

Another thing to note is that the uni-skip encoder and both the forward and backward encoder of the bi-skip vector, all of which focus on encoding a sentence and which can later be concatenated to construct the bi-skip and comb-skip vectors, seem to be biased to focus on learning either the syntax or semantics. The uni-skip representation has the highest correlation of all representations with the syntactic representations but scored the lowest on semantics. On the other hand, the two encoders of the bi-skip vector have the lowest correlation with the syntactic representation but have a relatively high correlation with the semantic representation. The combi-skip representation and the bi-skip representation, which in itself is a combination of the forward and backward encoded vectors, seem to perform well on capturing both syntax and semantics. This might indicate that encoders are vulnerable to being biased to learn either the underlying syntax or semantics and that the concatenation of those vectors leads to a fuller and less biased representation. In other words, by concatenating different learned representations, each of which encoded a

sentence from their own perspective as it were, a fuller representation can be built, which results into a better representation of the syntax and semantics of the original sentence. This explanation provides an answer for the other research questions.

7. Conclusion

This study evaluated the learned representations by RNNs on a more global level as opposed to other evaluations (e.g. Kádár et al., 2017; Linzen et al., 2016; Conneau & Kiela, 2018). More specifically, this thesis was focussed on investigating to what extent neural networks learn the semantics of sentences as opposed to the syntax. In order to answer this question, RSA was applied to pairs of neural representations and structured symbolic representations.

The results showed that neural representations capture the semantics of sentences to a larger extent than their syntax, a result which reflects the linguistic representations of concepts in languages thereby answering our main research question. In addition, the analysis proved useful for global evaluation of different representations. Moreover, the new approach introduced a new perspective on the representations learned by neural networks. More specifically, it suggests that combining vectors enables models to construct fuller representations which seem to be less biased to capturing either syntax or semantics, thereby answering the other research questions.

Although the novel approach of evaluating neural representations introduces a new perspective on these representations, further research is needed. Due to time and computational limitations, certain choices for the experimental set up were made which might influence the results. First of all, future research might compare the results using this approach to the performance on state-of-the-art benchmarks, such as SentEval (Conneau & Kiela, 2018) or GLUE (Wang et al., 2018). Moreover, the current study focuses on evaluating vectors made by only one model. The architecture of the model might influence the learned representations and results might differ for other models. In addition, the choices of the structured symbolic representations as well as their construction might also influence the results. Besides considering constituency trees as opposed to dependency trees to represent the captured syntax, future research might also use humanly constructed or verified trees. Finally, the corpus on which the representations were made consists of only one book. Qualitative analysis showed the corpus still contained at least one typographical error, specifically in sentence 361, which might influence the encoders. Due to time limitations, the corpus could not be further checked or corrected after the analysis. Future research

might consider larger and more diverse corpora and correct for possible errors. Despite these possible limitations of this study, the new approach still has proven useful for comparing neural representations and explaining their successful applications.

Another direction of future research might be to continue this work by examining the ratio of learned syntax and semantics. Can an optimal ratio be found, on which future models should be focussed to construct better representations? In addition, neural representations might be evaluated by comparing them to other representations as opposed to syntactic and semantic representations. This might lead to new insights about the neural representations from other perspectives as well. Finally, a comparison between various current and future state-of-the-art models might be made using this approach, which might inspire the construction of better models.

Acknowledgements

I would like to thank my supervisor, dr. Grzegorz Chrupała, for this very interesting and, in a positive way, challenging topic, and for all the guidance along the way. Your critical remarks sometimes made me question my whole thesis, yet you were always willing to help, which I think really improved not only my thesis, but also my scientific attitude and skills.

I would also like to thank Thiago Castro Ferreira for his quick adaptation of tree kernels so that I could use them for my dependency trees, and for his fast and elaborative replies, through which I quickly learned a lot about these kernels.

References

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Banarescu, L., Bonial, C., Cai, S., Georgescu, M., Griffitt, K., Hermjakob, U., Knight, K., Koehn, P., Palmer, M., & Schneider, N. (2013). Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse* (pp. 178-186).

Baroni, M., & Zamparelli, R. (2010, October). Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing* (pp. 1183-1193). Association for Computational Linguistics.

Belinkov, Y., Durrani, N., Dalvi, F., Sajjad, H., & Glass, J. (2017). What do Neural Machine Translation Models Learn about Morphology? *arXiv preprint arXiv:1704.03471*.

Bock, K., & Miller, C. A. (1991). Broken agreement. *Cognitive psychology*, *23*(1), 45-93.

Broere, B. (2018). *Syntactic properties of skip-thought vectors* (Doctoral dissertation, Tilburg University).

Cai, S., & Knight, K. (2013). Smatch: an evaluation metric for semantic feature structures. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)* (Vol. 2, pp. 748-752).

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Coecke, B., Sadrzadeh, M., & Clark, S. (2010). Mathematical foundations for a compositional distributional model of meaning. *arXiv preprint arXiv:1003.4394*.

Collins, M., & Duffy, N. (2002). Convolution kernels for natural language. In *Advances in neural information processing systems* (pp. 625-632).

Collobert, R., & Weston, J. (2008, July). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning* (pp. 160-167). ACM.

Conneau, A., & Kiela, D. (2018). SentEval: An Evaluation Toolkit for Universal Sentence Representations. *arXiv preprint arXiv:1803.05449*.

Conneau, A., Kruszewski, G., Lample, G., Barrault, L., & Baroni, M. (2018). What you can cram into a single vector: Probing sentence embeddings for linguistic properties. *arXiv preprint arXiv:1805.01070*.

Dyer, C., Kuncoro, A., Ballesteros, M., & Smith, N. A. (2016). Recurrent neural network grammars. *arXiv preprint arXiv:1602.07776*.

Elman, J. L. (1990). Finding structure in time. *Cognitive science*, *14*(2), 179-211.

Elman, J. L. (1991). Distributed representations, simple recurrent networks, and grammatical structure. *Machine learning*, *7*(2-3), 195-225.

Graves, A. (2013). Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Grefenstette, E., & Sadrzadeh, M. (2011, July). Experimenting with transitive verbs in a discocat. In *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics* (pp. 62-66). Association for Computational Linguistics.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735-1780.

Honnibal, M., & Johnson, M. (2015). An improved non-monotonic transition system for dependency parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing* (pp. 1373-1378).

Honnibal, M., & Montani, I. (2018). spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing. *To appear*.

Jordan, M.I. (1986). Serial order: A parallel distributed processing approach (Tech. Rep. No. 8604). San Diego: University of California, Institute for Cognitive Science.

Kádár, A., Chrupała, G., & Alishahi, A. (2017). Representation of linguistic form and function in recurrent neural networks. *Computational Linguistics*, *43*(4), 761-780.

Karpathy, A., Johnson, J., & Fei-Fei, L. (2015). Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*.

Kiros, R., Zhu, Y., Salakhutdinov, R. R., Zemel, R., Urtasun, R., Torralba, A., & Fidler, S. (2015). Skip-thought vectors. In *Advances in neural information processing systems* (pp. 3294-3302).

Kriegeskorte, N., Mur, M., & Bandettini, P. A. (2008). Representational similarity analysis-connecting the branches of systems neuroscience. *Frontiers in systems neuroscience*, *2*, 4.

Le, Q., & Mikolov, T. (2014, January). Distributed representations of sentences and documents. In *International Conference on Machine Learning* (pp. 1188-1196).

Li, J., Monroe, W., & Jurafsky, D. (2016). Understanding neural networks through representation erasure. *arXiv preprint arXiv:1612.08220*.

Linzen, T., Dupoux, E., & Goldberg, Y. (2016). Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *arXiv preprint arXiv:1611.01368*.

Logeswaran, L., & Lee, H. (2018). An efficient framework for learning sentence representations. *arXiv preprint arXiv:1803.02893*.

Matthiessen, C., & Bateman, J. A. (1991). *Text generation and systemic-functional linguistics: experiences from English and Japanese*. Pinter Publishers.

McCann, B., Keskar, N. S., Xiong, C., & Socher, R. (2018). The natural language decathlon: Multitask learning as question answering. *arXiv preprint arXiv:1806.08730*.

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

Mikolov, T., Yih, W. T., & Zweig, G. (2013a). Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (pp. 746-751).

Mitchell, J., & Lapata, M. (2008). Vector-based models of semantic composition. *proceedings of ACL-08: HLT*, 236-244.

Mnih, A., & Hinton, G. E. (2009). A scalable hierarchical distributed language model. In *Advances in neural information processing systems* (pp. 1081-1088).

Moschitti, A. (2006). Making tree kernels practical for natural language learning. In *11th conference of the European Chapter of the Association for Computational Linguistics*.

Moschitti, A. (2006, September). Efficient convolution kernels for dependency and constituent syntactic trees. In *European Conference on Machine Learning* (pp. 318-329). Springer, Berlin, Heidelberg.

Pennington, J., Socher, R., & Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (pp. 1532-1543).

Schütze, H. (1993). Word space. In *Advances in neural information processing systems* (pp. 895-902).

Schütze, H. (1998). Automatic word sense discrimination. *Computational linguistics*, *24*(1), 97-123.

Shieber, S. M., Pereira, F. C. N., Karttunen, L., & Kay, M. (1986). A Compilation of Papers on Unification-Based Grammar Formalisms, Parts I & II. Report 86-48, CSLI. *Stanford, Ca*, *80*, 239-243.

Socher, R., Huval, B., Manning, C. D., & Ng, A. Y. (2012, July). Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning* (pp. 1201-1211). Association for Computational Linguistics.

Socher, R., Pennington, J., Huang, E. H., Ng, A. Y., & Manning, C. D. (2011, July). Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the conference on empirical methods in natural language processing* (pp. 151-161). Association for Computational Linguistics.

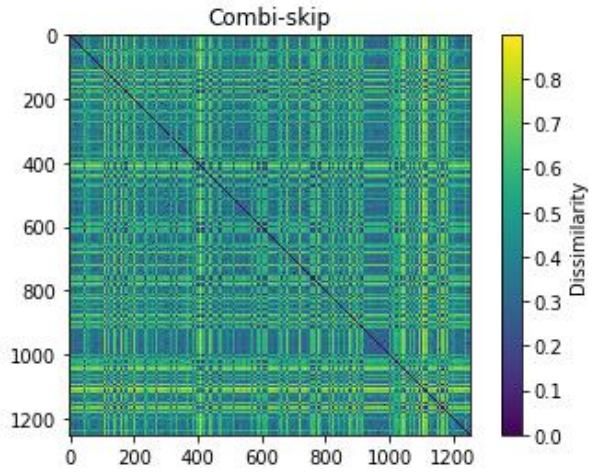
Turian, J., Ratinov, L., & Bengio, Y. (2010, July). Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics* (pp. 384-394). Association for Computational Linguistics.

Vinyals, O., Kaiser, Ł., Koo, T., Petrov, S., Sutskever, I., & Hinton, G. (2015). Grammar as a foreign language. In *Advances in Neural Information Processing Systems* (pp. 2773-2781).

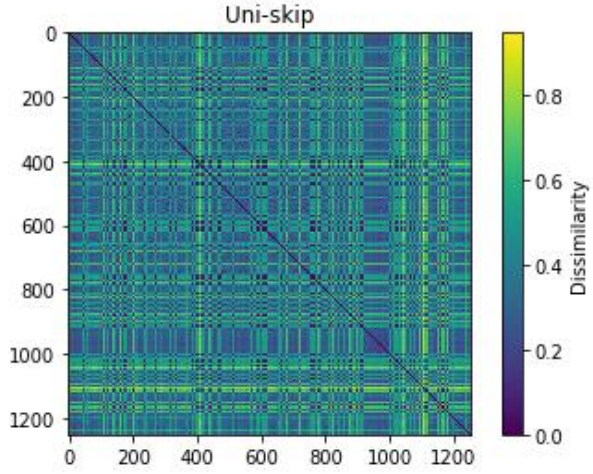
Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., & Bowman, S. R. (2018). GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. *arXiv preprint arXiv:1804.07461*.

Appendix A

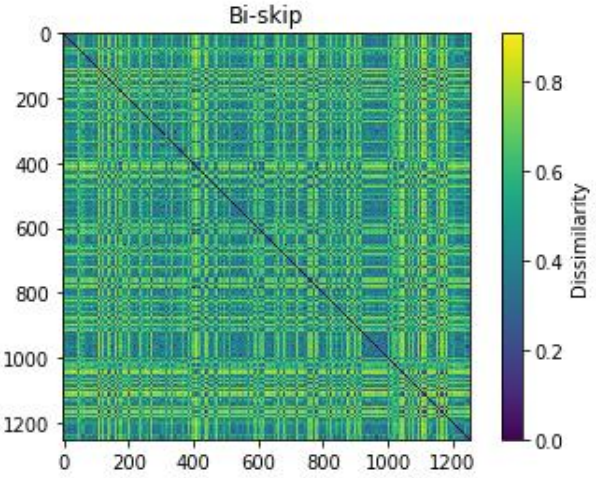
RDMs for the neural representations of sentences as represented by A) Combi-skip, B) Uni-skip, C) Bi-skip, D) Bi-skip forward, and E) Bi-skip backward.



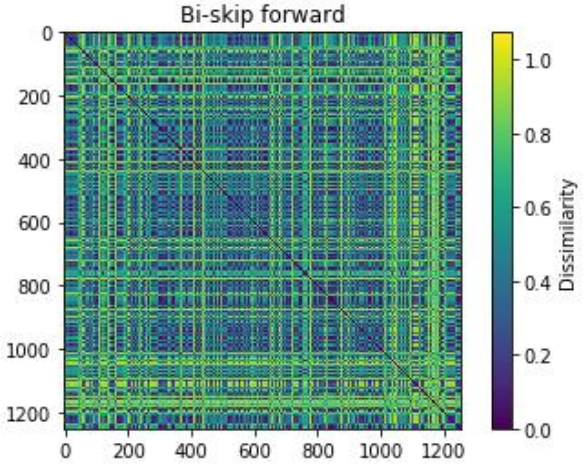
A)



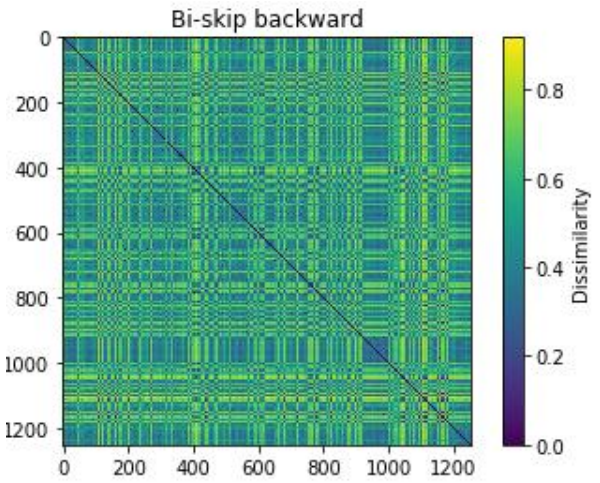
B)



C)



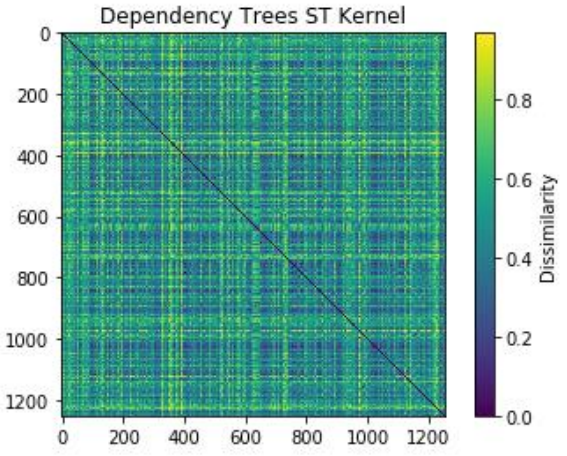
D)


Bi-skip backward
0
200
400
600
800
1000
1200
0
200
400
600
800
1000
1200
0.8
0.6
0.4
0.2
0.0
Dissimilarity

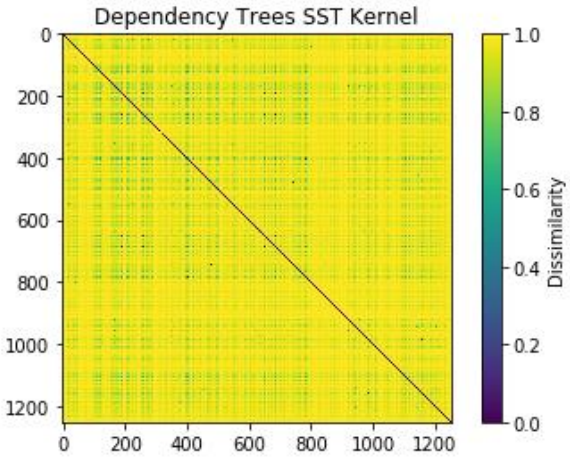
E)

Appendix B

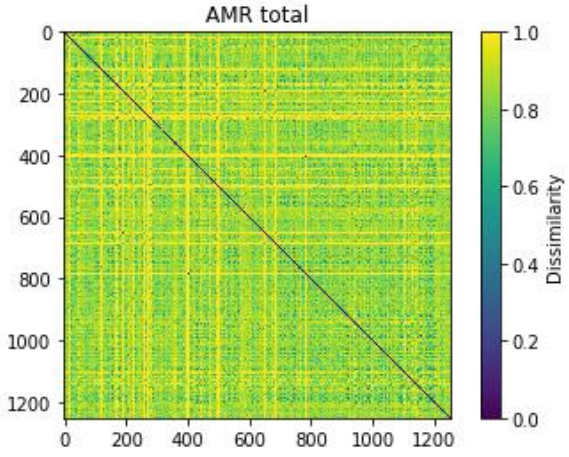
RDMs for the structured symbolic representations of sentences as represented by dependency trees evaluated using A) a ST kernel, B) a SST kernel, and AMRs evaluating C) all triples, D) instance triples, E) relation triples, and F) attribute triples.



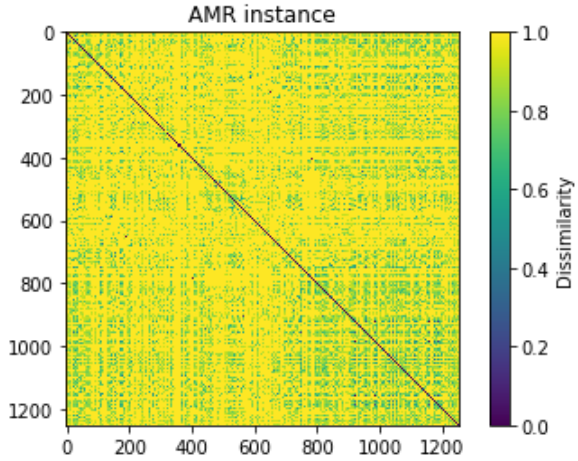
A)



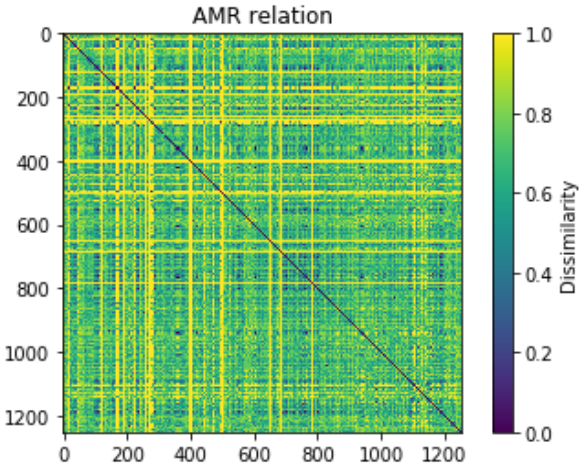
B)



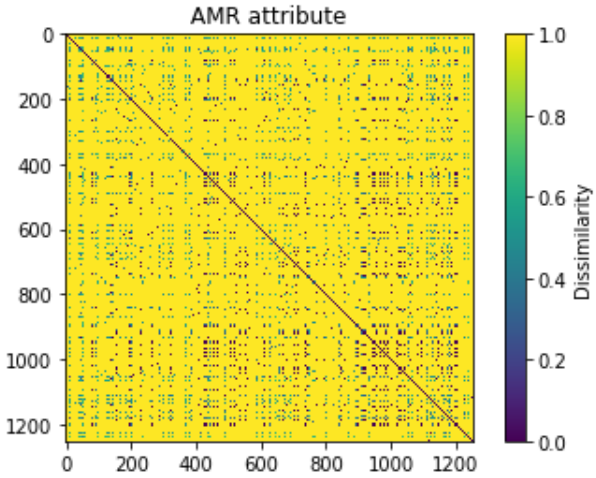
C)



D)


AMR relation
0
200
400
600
800
1000
1200
1.0
0.8
0.6
0.4
0.2
0.0
Dissimilarity

E)


AMR attribute
0
200
400
600
800
1000
1200
1.0
0.8
0.6
0.4
0.2
0.0
Dissimilarity

F)