University of Tartu

Institute of Computer Science

Computer Science Curriculum

Sander Nemvalts

Annotating Line Graphs With Machine Learning

Bachelor's thesis (9 ECTS)

Supervisor: Eduard Barbu

Tartu 2021

Joongraafikute kirjeldamine kasutades masinõpet

Lühikokkuvõte:

Selles bakalaureusetöös kogutakse andmestik, mis sisaldab joongraafikuid ja nende ingliskeelseid kirjeldusi. Kasutades seda andmestikku luuakse kaks erinevat masinõppesüsteemi, mis genereerivad joongraafikutele ingliskeelseid kirjeldusi.

Võtmesõnad:

Graafik, visualisatsioon, masinõpe, NLP

CERCS: P176 Tehisintellekt

Annotating Line Graphs With Machine Learning

Abstract:

In this thesis, a dataset of line graphs and their descriptions is assembled. Using that dataset two different machine learning models are constructed that are able to generate natural language descriptions of line graphs in English.

Keywords:

Graph, plot, figure, visualization, machine learning, NLP

CERCS: P176 Artificial intelligence

Table of Contents

1. Introduction

Graphs, plots, figures and data visualizations are used to better prepare numerical data for human consumption. Plots make data much easier to understand for humans, and natural language descriptions added to those plots can improve comprehensibility of the data even more. These descriptions can highlight important trends and relationships in data, which might not be noticed otherwise. For visually impaired people, these descriptions are the only way they can "see" the visualization.

Natural language descriptions are usually written by humans. Writing the description requires understanding of the visualization, concise writing skills, and a fair bit of effort. These qualities are hard to replicate with computers, requiring complex image recognition systems for interpreting the images and natural language systems to generate the description. By automating this process using machine learning, it can be shown that it's possible to extract key details from images of visualizations to generate concise natural language summarizations. These generated summarizations can later be attached to the visualizations, saving time and effort. Since these summarizations are not written by humans, in theory they could also offer a different point of view to the visualization, and could highlight hidden details and relationships in the data.

In this thesis, image and figure captioning concepts are first introduced with the architecture relevant to them, along with in-depth descriptions of some previous work in figure captioning. Then, a figure captioning dataset generation process inspired by previous approaches is presented. Finally, two structurally different figure captioning models are described and evaluated and some recommendations are presented for future work.

2. Image and figure captioning

In this chapter, basic image captioning concepts are first examined, since the field is closely related to figure captioning. Figure captioning and its concepts are then examined, along with some exemplary approaches for each concept.

2.1 Image captioning with machine learning

Image captioning problems are machine learning problems, where the objective is to generate a natural language description that describes a source image. To achieve this, the final system must be able to "read" and recognize relevant parts and patterns of the image, and then transform those patterns into the final result - a natural language description of the image. Since all figures and plots are images, figure captioning is a subproblem of image captioning [1]. General image captioning has also been explored more than figure captioning specifically, which results in figure captioning solutions applying concepts from image captioning [1].

2.2 Architectures relevant to image captioning

Image captioning systems are often built with the architecture of encoder-decoder**.** Both the encoder and decoder have clearly specified roles. The encoder is responsible for reading and recognizing the relevant parts of the image. It outputs an intermediate representation (often a matrix of values) that summarizes the image, albeit not in a directly human-interpretable form [2:469]. This intermediate representation is then inputted into the decoder, that is responsible for converting the intermediate representation into the final product, a description [3:298]. A summary of the architectures and components used for image captioning is presented in Table 1.

Table 1. A summary of primary components used for image captioning.

Component	Usage of component	Output of the component
Convolutional neural net (CNN)	Used to transform the image into a intermediate representation, using convolution and pooling operations	A matrix of real numbers that summarizes an image
Categorical decoder	Used to transform the intermediate representation of the encoder and any other information into a discrete category of the image in image classification problems	A discrete category of an image, such as "dog", "cat", or "car"
RNN based decoders	Used to transform the intermediate representation and any other information into a caption that is generated word-by-word	A word-by-word output that describes an image

The structure of the encoder can vary for image captioning tasks, but in most cases it is a convolutional neural network [3:298]. For tasks where images are processed, one of the most important layer types in the deep neural network of the encoder is the convolutional layer. The convolutional layer is a neural network layer that uses a linear convolution operation on the pixels of an image [2:327]. This enables the encoder to recognize patterns in the image, rather than focusing on the values of the pixels themselves. The convolutional layer is later followed by a pooling layer that helps the model generalize the location of the features of the image [2:336]. A varying combination of these layers forms a deep neural network that is suitable to be used as an encoder.

For problems related to images, the architecture of the decoder varies more than the encoder's, and is dependent on the pattern and structure of the final result. When captioning images with a single word or type, the entire problem can be formulated as a classification problem. For classification problems, where the output is a discrete value of a category or a type, using a Softmax layer on top of the encoder to output the probabilities for all the different description categories can suffice [2:180-181].

However, when captioning an image with a multi-word description, a more complex architecture is needed. The model designed with this architecture should be able to form and find connections in data that is sequential in nature (such as sequences of words) to help it achieve correct syntax when generating a natural language description. To achieve this, Recurrent Neural Nets (RNNs) are used [2:386]. RNNs are a class of neural networks that are characterised by some form of

recurrence occurring in the network, being able to refer to either previous outputs, or previous hidden states. This makes RNNs well suited to processing sequential data, such as natural language descriptions [2:367].

One of the earliest and most common RNNs used for image captioning problems are Long-Short-Term-Memory (LSTM) cells [4] [5]. The most important component of the LSTM cell is the cell state, which can store information on all the previous inputs the recurrent model has seen, and "forget" information in the cell state with the forget gate [5]. With the cell state and the forgetting mechanism, the LSTM is well suited for image captioning tasks [2:404].

2.3 Previous work related to figure captioning

In previous approaches to figure captioning, both the figures and the data the figures have been constructed from have been used to generate natural language descriptions [1, 6, 7]. To generate the final natural language captions, both machine-learning and rule-based systems have been used.

2.3.1 Rule-based approaches to figure captioning

Rule-based approaches to figure captioning have been one of the earliest approaches to figure captioning. They are characterized by a complex manual system that employs rule-based matching. These systems don't leverage machine learning, and rely on the manual system to select relevant statistics from the figure and to convert it into some intermediary form. This intermediary form is later used with rule-based systems to fill in text templates and to generate the final description. The captions generated by these systems are often synthetic and don't explain the graph in-depth [7].

In Demir et al [8], a classic rule-based figure captioning system is described for bar charts. To generate a caption, the system first extracts the elements of the image as XML. The representation along with manually inferred messages of the graph is then inputted into a system. The system ranks the messages based on their importance, as perceived by the volunteers who tagged them.

These messages are inputted into another system, where they are grouped and classified according to rules, and are then used to form candidate trees that describe potential description forests. The forests are then evaluated with a custom evaluation system, taking into account

sentence count, syntactic complexity, and comprehension complexity. Another system orders the description structures to optimize for comprehensibility. Then, the trees are converted to final descriptions using a syntax realizer. The generated descriptions describe the bar charts well, albeit a bit synthetically.

As evidenced by the description of a typical rule based system, rule based systems are very complex. They require large amounts of manual work and programming to capture the essence of the chart that is later used to generate descriptions.

2.3.2 Machine learning and templating approaches to figure captioning

Compared to rule-based approaches to figure captioning, approaches using machine learning tend to be less complex and easier to comprehend. As with image captioning, the machine learning approaches to figure captioning tend to have a structure of an encoder and a decoder. The encoder generates some features based on the image of the figure that is later transformed into a description or a caption using a decoder. The decoder can be a simple templating system [6], or a more complex machine learning system that outputs descriptions word by word [1, 7]. With a templating system, the machine learning system predicts a trend and values that follow that trend. Based on the trend, a pre-written template is picked. The values are replaced into the pre-written template, resulting in a concise and grammatically correct caption [6]. For example, if the machine learning system predicts the trend "GREATER" and values "Red" and "Blue", and a template "A is greater than B" is picked based on the trend, then after replacing the values into the template the final caption will be "Red is greater than Blue".

One of the most recent approaches to figure captioning with machine learning and templating is described in Liu et al [6]. The dataset used in that approach consists of charts in the scalable vector graphics (SVG) format, where the elements used to construct the chart (lines, rectangles, text) are easily accessible. These elements are collected from the graph and are parsed and grouped with a multilayer perceptron (MLP) classifier. Then, the elements are inputted into a chart feature extractor, which extracts human-interpretable features about the chart. Finally, these features are used with manually created templates to fill in the chart description [6]. This approach results in reliable and syntactically correct captions and basic descriptions of the graph. However, since a templating system is used, this often results in synthetic and repetitive descriptions, as noted by the volunteers recruited to rate the generated descriptions [6].

2.3.3 Machine learning approaches to figure captioning

Compared to templating based systems, purely machine learning solutions don't rely on a final templating system to generate descriptions. Rather than using a templating system to construct the final captions, the purely machine learning approach uses a decoder that generates the description word by word. This approach places more emphasis on the natural language part of the problem, since like in image captioning problems the decoder is required to learn syntactic structures of sentences to generate grammatically correct descriptions [3:278] [3:299], while still having to capture the essence of the image.

One of the latest examples of this is the approach presented in Chen et al [1], where the final system generates captions word by word for synthetic figures. The synthetic dataset has been generated from a question-answer (QA) dataset named FigureQA [9], using ground-truth data available about the figure (the values that were plotted to produce the figure) and the generated figure itself. The generated figures are bitmap images, which means that elements of the graph are harder to recognize and represent compared to vector graphics.

To encode the bitmap images of the figure, a ResNet is used [1]. This produces feature maps that encapsulate patterns of the image. These feature maps are used with a relation attention network, to produce a relation map between each feature map. Additionally, an attention system using embeddings of the labels in the figure is used. These three components (feature attention, relation attention, label attention) are used to construct a context vector. The context vector is inputted into the LSTM decoder, that generates the description word-by-word. With all of the attention systems enabled the model performs well, albeit only for shorter captions [1].

In purely machine learning approaches, only the data used to construct the graphs can be used for figure captioning as well. This is the approach taken in Obeid et al [7], using the Transformer model to generate large natural language descriptions. The dataset used to train this model are graphs and long natural language descriptions from the statistics website Statista. A novel subject-replacement system is introduced. The subject replacement system replaces the subjects in the descriptions with replacement words, to help it generalize. For example, in the description "Red is greater than Blue", the subjects "Red" and "Blue" would be replaced with placeholders. The final outputs of the model are very natural, complex and resemble the source captions.

3. Methodology

In this chapter, two different figure captioning datasets are introduced, along with the process required to generate the first one. Two different figure captioning models are also presented, as well as the process of training them.

3.1 Dataset

To train the model, a custom dataset is introduced. It is based on the FigureQA [9] question-answer dataset, and the approach used to generate captions is similar to the method used to construct FigCAP, specifically the detailed variant (FigCAP-D) [1]. The code to generate the dataset is available in a GitHub repository[1].

The dataset used for this model consists of line graphs, the names of the values plotted within them, and descriptions of those graphs. Most often, line graphs are used to show a change of one or more values over time. An example of a line graph is shown in Figure 1.

[1] https://github.com/snemvalts/line-chart-captioning

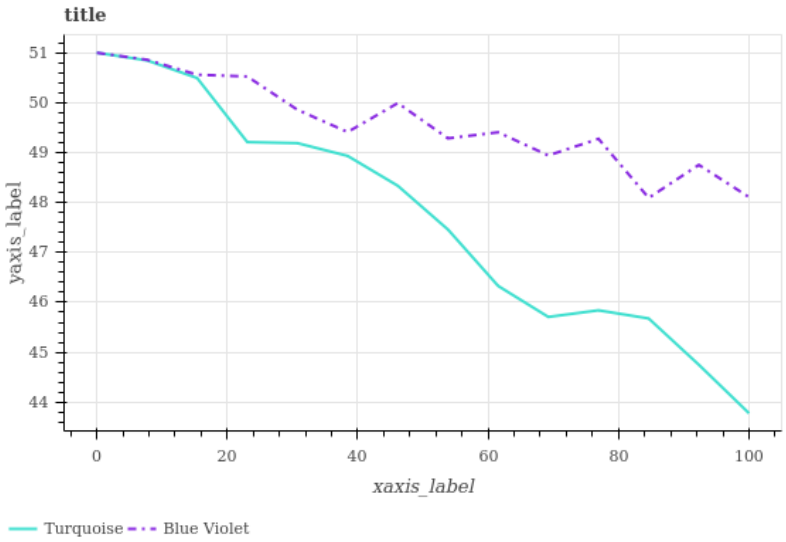

Figure 1. An example of a line graph from the FigureQA dataset. The values plotted are "Turquoise" and "Blue Violet". A single natural language description is associated with this graph: "Blue Violet intersects Turquoise" [9].

For every line graph in the generated dataset at least one description of the graph exists. This description can be in a natural language form, or in a template form, with the type of description and names of subjects provided. For Figure 1, a figure from the FigureQA dataset, a single description exists: "Turquoise intersects Blue Violet" as a natural language description, or as (INTERSECTS, "Turquoise", "Blue Violet").

On a high level, the dataset is constructed as follows:

1. First, the FigureQA dataset is loaded and processed.
2. Then, line charts are filtered out, discarding all other types of charts present in FigureQA (such as bar and pie charts).
3. After that, the question-answer pairs are loaded. A question-answer pair contains a reference to the image of the figure, the type of the question, and the subjects the question is about.
4. Question-answer pairs with "true" as the answer are filtered out and formed into descriptions. The descriptions and the subjects portrayed in the chart are attached to the image of the chart.

FigureQA includes images of the figure, as well as the data used to construct that figure. The images included in the dataset are used as images of the figures, and only the list of subjects is

extracted from the data used to construct the figure. For example, in Figure 1 the list of subjects extracted would be ["Turquoise", "Blue Violet"].

As for extracting question-answer pairs, FigureQA includes 9 different question types related to line charts ("MIN_AUC", "MAX_AUC" "SMOOTHEST", "ROUGHEST", "GLOBAL_MIN", "GLOBAL_MAX", "LESS", "GREATER", "INTERSECT"). Question types "LESS", "GREATER", "INTERSECT" are associated with two subjects, and form a question about how two different subjects relate to one another. For example, question type "LESS" with subjects "A" and "B" forms a question: "Is A less than B?". If the type were "GREATER", the question would be "Is A greater than B?". With "INTERSECT", the question would be "Does A intersect B?".

All other question types only involve one subject, and are statements about an outlying subject in the graph. For example, question type "MIN_AUC" with subject "A" forms the question "Does A have the smallest area under the curve?", question type "SMOOTHEST" forms the question "Is A the smoothest value?" and "GLOBAL_MAX" forms the question "Does A have the largest value?".

The desired question-answer pairs are filtered to only include desired question types that are about the filtered line graphs. These desired question types can be set with a JSON configuration file. By default, the question types "GREATER", "LESS" and "INTERSECT" are used.

The answer portion of the question-answer pair can be either "Yes" or "No". After filtering for desired question types, only question-answer pairs with the answer "Yes" are kept. These questions with the answer "Yes" are then interpreted as descriptions of the graph, since the questions can be formed into a statement. This forms the basis for the ground-truth descriptions used to describe the figure.

After obtaining the image of the figure, the list of subjects and the true question-answer pairs, templating is performed. Given a question with the answer "Yes", the question is inserted into a template to form a natural language description. For example, given the description in Figure 1 (["INTERSECT", "Turquoise", "Blue Violet"]) and a template "A intersects B", a description "Turquoise intersects Blue Violet" is formed. A variety of templates can be provided to increase the diversity of descriptions. This is done in the same JSON configuration file used to provide the desired question types.

By default, the sentences associated with a certain graph are then concatenated to form a combined description describing the figure. For example, if a figure with the identifier "932" has 2 question-answer pairs (and thus sentences) associated with it, such as "Blue is greater than Red" and "Purple is less than Red", the resulting combined description of figure "932" will be "Blue is greater than Red. Purple is less than Red."
After the combined description is associated with the graph, the processing is complete.

3.2 Dataset augmentations

To accommodate different approaches and experiments with models, a variety of augmentations can be enabled when generating the dataset. All of the augmentations augment the description, the images are not changed. Each of the augmentations is optional, and are enabled with command line flags for the augmentation script. Some of the augmentations cannot be enabled together at once, the dataset processing is aborted when run with incompatible augmentations enabled.
The most basic augmentation is the description limit. This limits the length of the combined descriptions to n, where $n > 1$. This can be used to better balance the length of descriptions in the dataset, as some figures have more descriptions than others.
The description unrolling augmentation can be used to reduce the length of combined descriptions to one. With this augmentation enabled, combined descriptions are split apart. For example, if a graph with the identifier "602" normally has a combined description with 3 sentences, this combined description is split apart into 3 entries. This means that the graph "602" will appear 3 times in the unrolled dataset, once each with one of the descriptions. This lowers description length and can help the performance of the model.
Subject replacement, inspired by Obeid et al [7], replaces the subjects in the descriptions, to help the model generalize. For example, a combined description "Blue intersects Red. Purple is greater than Red" becomes "<A> intersects <B>. <C> is greater than <B>" with subject replacement. A subject replacement map is associated with the description as a JSON string of a key-value store. The key-value store maps each replacement to the underlying subject, which can be used to reconstruct the original caption later.
When replacing subjects and the description unrolling augmentation is enabled, "local" subject replacement can also be enabled. This ensures that only the captions present in the unrolled

description are taken into consideration when replacing subjects. On a high level, it means that only "<A>" and "<B>" can refer to subjects in unrolled descriptions.

The final augmentation that is available is stored in a separate script, and it skips the templating part of the dataset generation. This means that the descriptions generated will only contain the type of the question and the subjects involved. The "description-types" dataset generated with this script is used for the baseline model that requires discrete values rather than natural language descriptions.

To generate the training set specifically, a pre-assembled training split for FigureQA [9] is used. A pre-assembled test split is used for the test set. The final training set contains around 19000 figures and around 45000 questions about those figures. The test set contains around 3800 figures and 9125 questions about those figures.

3.3 Alternate dataset

The second dataset is a non-synthetic "natural" dataset, that is based on the work of Obeid et al [7]. The dataset is scraped from the statistics website Statista and consists of images of the graph, data used to construct the graph, and the natural language descriptions. The dataset was considered and evaluated, given the impressive results of the model described in Obeid et al, and the fact that very few "natural" and handwritten datasets are present for figure captioning [7].

To create the dataset, some processing work is still required, albeit less than for the primary dataset. To only get line graphs from the dataset, the graphs are filtered again, excluding any other chart types. The line graphs themselves in the dataset are only time series graphs, denoting change in some value over time [7]. Every line graph has two different images associated with it. One of the images is a screenshot of the Statista web page containing the plot, and the other is a regenerated figure image that has been generated using Matplotlib and the scraped data used to construct the original figure. The screenshot images of the figure contain unrelated information from the webpage [7], so the regenerated figure image is used. For some of the regenerated images present in the dataset provided as part of the publication, axes are flipped and the figure is incorrect. To exclude these, a list of incorrect images had to be manually constructed.

For the list of subjects portrayed in the graphs, the headers of the underlying data tables are retrieved.

The descriptions provided are used as-is. The descriptions have been written by hand, so they form meaningful and coherent texts, as opposed to synthetically generated descriptions. This means that the average length of a description is much longer: the average in the alternate dataset is 113 words or 9 sentences, compared to around 13 words or 2 sentences in normal descriptions and 5 words or 1 sentence in unrolled descriptions.
During the development of the models on the synthetic dataset, it was chosen not to use the alternate dataset, as the hand-written descriptions would be hard for the recurrent model to predict, purely based on the image of the figure.

3.4 Models

To create machine learning models capable of figure captioning, two different model architectures are presented: a baseline model and a recurrent model. Both of the models have been designed to work on the generated synthetic dataset and generate captions for figures. The models have been written in PyTorch. The Jupyter notebooks used to build them are available in the same GitHub repository as the dataset.

3.4.1 Baseline model

The baseline model is based on the "machine learning and templating" approach to figure captioning. Given a bitmap image of a figure and the list of elements contained within them, the baseline model generates a description type and predicts two subjects that interact in this way. The description types are identical to the question types presented in chapter 3.1. The description type is later matched to a template, the two subjects are filled in, and the description is generated. Figure 3 denotes an example of this prediction. The first model receives an image of the figure and the subjects in the figure, predicts ["GREATER", ["Orange", "Tomato"]], and this is manually templated to form the caption: "Orange is greater than Tomato".

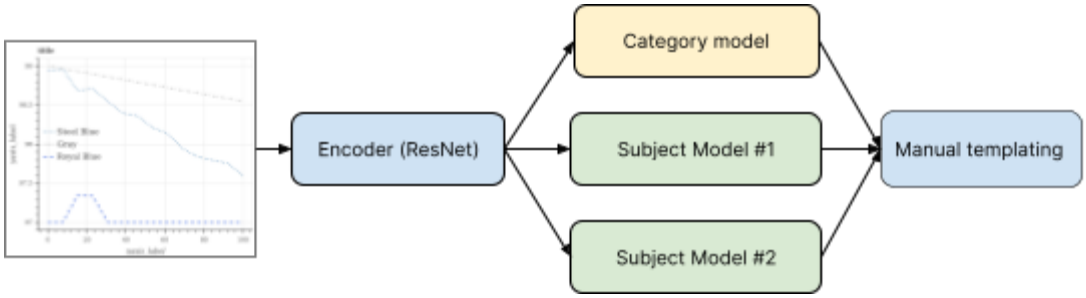

Figure 2. An overview of the architecture for the baseline model. The figure is inputted into the encoder, and the encoded image is inputted into the category and subject models. The predictions from the category and subject models are then manually templated to form a complete caption

The baseline model consists of four different models: the encoder, the category model and two subject models. The architecture for the baseline model is illustrated in Figure 2. The encoder encodes the image into a matrix representation that aims to encapsulate the patterns and features present in the image. The encoded image is inputted into the category model, which predicts the description categories present in an image. The encoded image, the predicted category and the other subject involved are passed into both of the subject models, and the subject models predict which other subject could have interacted in this way. The predicted category, the two predicted subjects are combined to form the description. In the example given with Figure 3, the category prediction model predicted "GREATER", the first subject model predicted "Orange" and the second subject model predicted "Tomato".

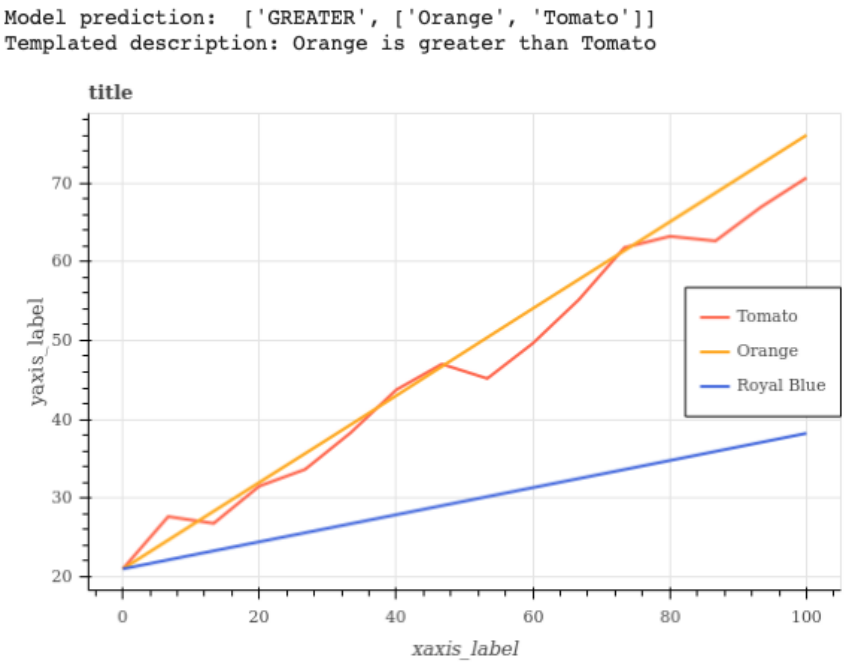

Figure 3. An example of a prediction by the baseline model. A category, and two values are predicted. They are templated to form a complete description.

The encoder used for this is an off-the-shelf ResNet [10], with the last category prediction layer removed and the second to last pooling layer replaced. The ResNet itself is a deep image classification network, that is notable for including "short connections", which can bypass some layers. This skipping helps with vanishing gradients and "degradation", where accuracy saturates and then starts decreasing [10]. Figure 4 depicts the architecture of a typical ResNet.

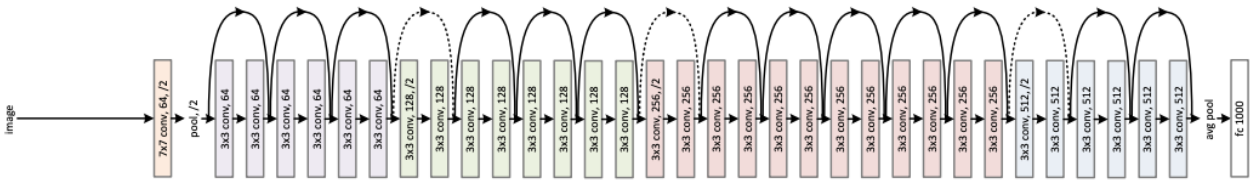

Figure 4. The architecture of a typical ResNet. The network is characterised by its depth and the utilization of skip connections between layers, with dotted skip connections showing an increase in dimensionality [10].

Both pre-trained and not trained ResNets can be used, and fine-tuning can be enabled for the pre-trained model. Either one of ResNet-18 and ResNet-50 (that are 18 and 50 layers deep respectively) can be used for the encoder. After the image is passed into the ResNet, the result is

average pooled. This results in a large matrix that encapsulates features and patterns present of the image of the figure.

This encoded image is inputted into the category model. The category model is a deep neural network with 6 hidden layers. The activation function used for the hidden layers is ReLU. After the encoded image has been propagated through the hidden layers, an output layer coupled with the softmax activation produces the probabilities that for every description category, the description category is present in the figure. A category prediction is defined as positive when the probability of a given category is above $\frac{1}{n}$, where n is the amount of description categories present in the dataset.

The subjects model works in a similar fashion. The subjects model is inputted the encoded image, a description type (as a one-hot vector), the amount of subjects present in the image (as a vector with a length of *n*, where *n* is the maximum amount of subjects present in the dataset), and the "opposite subject" (as a one-hot vector). The opposite subject is the subject that interacts with the subject being predicted. For example, in the example prediction presented in Figure 3, the opposite subject for "Orange" would be "Tomato", and the opposite subject for "Tomato" would be "Orange". If there were any more subjects portrayed in the figure, these opposite subjects would remain the same.

The encoded image, the one-hot vectors for description category and the opposite subject are passed into input layers for each of them, and they are then concatenated to form an input matrix, that is passed to the hidden layers. There are five hidden layers, and the activation function used is also ReLU. The output of the last hidden layer is passed to an output layer that produces probabilities with Softmax activation, very similar to the category model.

The softmax predictions are modified further programmatically. First, probabilities that were beyond the provided subject length are set to 0. For example, if the maximum subject length (that is present in the dataset) is 8, but the figure only has 3 subjects present, the last 5 probabilities are set to 0. The prediction score for the opposite subject is also set to zero.

There are two of these subject models, one for the first subject and the second for the second subject. After both the category and the subjects have been predicted by both of the models, all of the predicted values are combined and templated to form a natural language description about the figure.

3.4.2 Recurrent model

The recurrent model is based on the "purely machine learning" approach to figure captioning. The result is an end-to-end model that consists of an encoder, and a LSTM-based decoder with a feature attention system. This means that compared to the baseline model, there is only a single decoder for generating captions, without separate models for predicting subjects and description categories. The decoder also generates captions word-by-word, compared to the baseline model which only generates the category and subjects to be templated to form a description. This means that the decoder needs to learn the syntax and structure of descriptions as well, not just the reasoning required to describe a figure. The overall architecture for this model is inspired by the baseline presented in Chen et al [1] and is depicted in Figure 5.

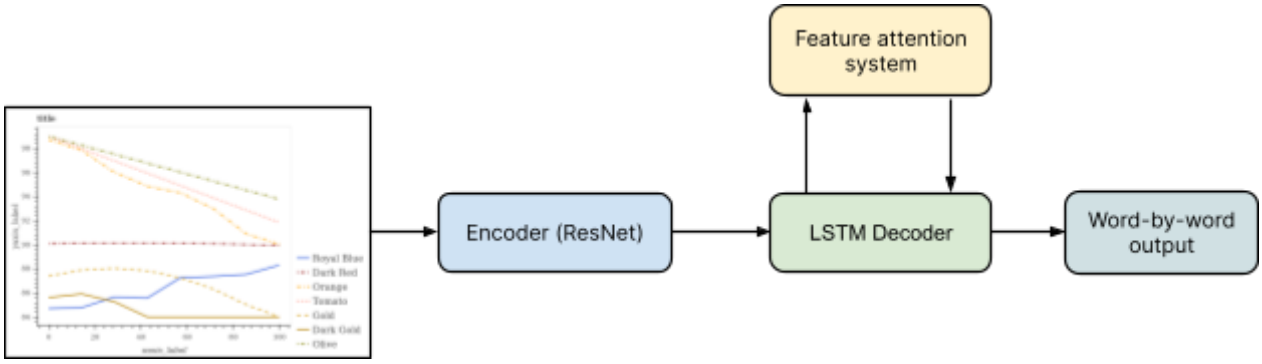

Figure 5. The architecture of the recurrent model. The image of the figure is inputted into the encoder, the encoded image is inputted into the LSTM decoder. The LSTM generates the description word-by-word using the feature attention system.

To first read and encapsulate patterns from the image an encoder is used that is identical to the ResNet used in the baseline model. This produces an encoded image, that is inputted into the decoder. The decoder consists of a LSTM for decoding the captions, and a feature attention system to help the model pay attention to different parts of the image.

When the encoded image is inputted into the decoder, an initial hidden state and cell state is computed. After the initial state is set up, tokens are generated *n* times, where *n* is the maximum length of the caption in the dataset. For every token that is predicted, the attention-weighed encoded image is calculated. This is done by inputting the hidden state of the LSTM and the encoded image to the feature-attention system.

The feature attention system has two parallel input layers, one for the encoded image and the other for the hidden state of the LSTM. After both the encoded image and the hidden state have been passed through their respective input layers, they are concatenated and passed through the ReLU activation function. Then, the concatenated matrix is passed through a linear layer. The output of the linear layer is fed into the Softmax activation function, to calculate coefficients of attention in the range of 0 and 1, with the coefficient 1 meaning that the corresponding vector is a very important part of the encoded image, and 0 meaning it's not important at all. The attention coefficients are then multiplied with the encoded image to produce an attention-weighted encoded image.

After the attention-weighted encoded image is calculated, the context vector is formed. The context vector is the feature attention vector by default, but a relation attention system similar to Chen et al [1] can be used as well. When using multiple attention systems at once, the context vector is formed by concatenating the output of attention networks (such as feature and relation attention).

The context vector can then be concatenated with either the embeddings of ground-truth captions, or the previous predictions to form the final input to the LSTM. Teacher forcing can be used by replacing the previous predictions with the embeddings of ground-truth captions when forming the context vector. The resulting matrix is inputted to the LSTM along with the current hidden and cell state, and a new hidden and cell state are computed. Finally, predictions are extracted from the hidden state with an output layer and the Softmax activation function.

3.5 Training the models

The models were trained on the Google Colaboratory platform, which provides a Python environment along with free access to compute hardware relevant to machine learning (GPUs and TPUs). Both of the models have been trained on the primary dataset (the one inspired by FigCAP), although with different augmentations.

All of the models used in the baseline model were trained with the augmentation that skips templating the question, since the objective of the baseline model is to only predict description categories and subjects. For the encoders, a pretrained ResNet-50 trained on the ImageNet dataset was used. Beyond that, different sets of augmentations were used depending on which model of the baseline model was being trained. For the subject prediction model, no additional

augmentations were enabled. For both of the subject models, the description unrolling augmentation was enabled. This was done because of technical reasons within the data-loading system of PyTorch, and to save time with a "good enough" solution.

The recurrent model was trained with the description unrolling and local subject replacement augmentations enabled. The description unrolling was enabled for the same technical reason as with the baseline subject models. The local subject replacement augmentation was enabled to help the model generalize, so that the model wouldn't have to predict specific names of values present in the image ("Tomato", "Blue"), and could just predict "<A>" or "<B>". This was further inspired by the fact that the recurrent model won't receive any information about the values present in the figure, except for the legend present in the image (the box in the figure that lists the values depicted in the figure along with the respective colors).

Both of the models were trained for 2-3 epochs on the entirety of the training data. The training took a total of 3-4 hours for each of the models. The training was stopped early because of the models no longer improving - loss and accuracy would remain very static, indicating underfitting. Experimentation with hyperparameters (mainly learning rate, network size and batch size) had small improvements on the loss and accuracy. When training the recurrent network with teacher forcing enabled, similar underfitting is experienced, until a certain point where the model starts predicting captions identical to the ground truth captions used for teacher forcing. This results in heavy overfitting on the training dataset, indicating that the model has "learned" to use the embeddings of the ground truth captions. Since these ground truth captions aren't provided when evaluating the model, teacher forcing was not used when training the final version of the recurrent model.

4. Results

In this chapter, the two models are evaluated with different metrics, and the findings from the evaluations are discussed. Some potential improvements for similar models are also proposed.

4.1 Baseline model results

Overall, results from the baseline model were moderately promising. While both the subjects and category model didn't train very well (loss and accuracy improved and then leveled off quite quickly), the model can still produce good descriptions of line figures.

One of the most straightforward ways to evaluate this model is to check if, given a figure, the model predicts the ground truth descriptions that are associated with the figure. However, it is important to note that this method isn't perfect: a certain figure can have many more possible descriptions than is present in the test dataset. This makes the metrics generally used for classification problems (such as accuracy, recall and f-score) not as concrete and indicative of model performance as for normal classification problems.

Three different variations of this accuracy metric are used. In the first variation, a correct prediction occurs when any predicted top-k description appears in the ground truth descriptions. In the second variation, a correct prediction only occurs when a description directly matches a ground truth description. The third variation is identical to the second one, except for the fact that instead of using the category predictions from the model, ground truth description categories are used for predictions in the model.

The accuracies of the model according to the different evaluation variations are depicted in Figure 6. The first variation naturally has the highest accuracy, since it is the most forgiving: only one description needs to appear in ground truth descriptions. The second and third variation have a naturally lower accuracy, but the accuracy of the third variation is higher than the accuracy of the second variation. Since the third variation is identical to the second variation, only difference being that it skips the category prediction, this suggests that the category prediction model might not be ideal, since it is causing a considerable gap between the accuracies.

The "random guess" accuracy baseline is also depicted in this figure. The formula for the baseline accuracy is $k \cdot (\frac{1}{n} \cdot \frac{1}{i} \cdot \frac{1}{i-1})$, where n is the amount of description types, i is the average amount of subjects present in the dataset and k is the k in top-k. For the dataset used with the baseline model, $n = 3,\ i = 4$.

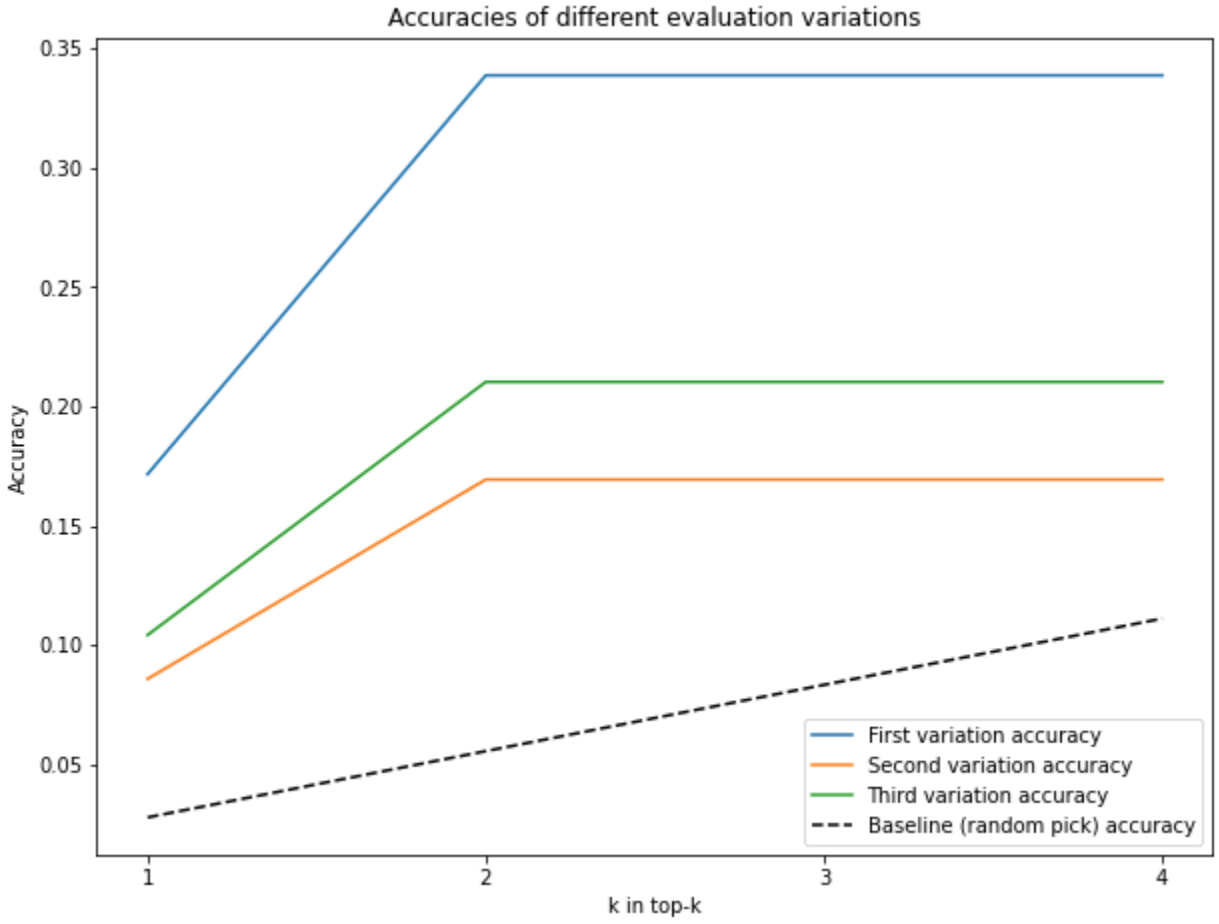

Figure 6. Accuracies of the model with different evaluation variations, given how many descriptions were considered from the descriptions generated. A baseline accuracy is also included.

As for a more high level evaluation: overall the model produces descriptions that describe the figure adequately. In some cases, both of the subjects models predict the same value, resulting in a nonsensical description ("Red is greater than Red"). The top-2 output of the subjects model also tends to be the top-1 description, but in reverse. For example, if the top-1 description would be "Red is greater than Blue", then the top-2 description would be "Blue is greater than Red".

Also, the categories model predicts the "INTERSECT" description type very infrequently, if at all.

Overall, the results aren't close to ideal, but as seen with the evaluation metrics, the model can produce descriptions that accurately describe the figure.

4.2 Recurrent model results

With the recurrent model, the results were far from ideal. The model learned the general sentence structure of the descriptions, by adding sentence start and end tokens, but most of the time fails to produce a caption that accurately describes the figure, not including any relational words (such as "greater", "lesser", "intersects"). When teacher-forcing is enabled, the model produces very good captions during training time, but performs very poorly during validation, indicating overfitting on the embeddings of the target captions. With a much lower learning rate and with teacher forcing enabled, the model still fails to generalize and produce any meaningful captions.

With the recurrent model, more standard metrics can be used, since the model also needs to learn the sentence structure and syntax. On the test dataset, the model achieves an average BLEU-3 [11] score of 0.27, and an average top-2 accuracy of 62%.

4.3 Comparisons with previous approaches

The baseline model is difficult to compare with the primary previous approach, Liu et al [6], given the lack of metrics and the open-endedness of the descriptions. As mentioned previously, a description could correctly describe a figure, but not be part of the ground-truth descriptions that are known to be true about the figure. Text generation and accuracy metrics such as BLEU [11] would be unsuited as well, because most of the final output would be a pre-written template, where most of the words are already aligned, and the structure and syntax of the sentence is correct. To compare the performance of the baseline model to Liu et al [6], a questionnaire system would have to be created and objective volunteers recruited to grade these predictions. The scope of that task would far eclipse the scope of this bachelor's thesis. The comparison would be complicated further by the types of figures captioned by the model: the Liu et al [6] system can generate descriptions for bar charts, line charts and scatter plots, while this baseline model could only generate descriptions for line charts.

For the recurrent model, the results are more comparable with the corresponding approach, Chen et al [1]. The most comparable dataset type is FigCAP-D, that includes detailed descriptions about the figure. The most comparable model type is CNN-LSTM+*Att_F*, the LSTM based decoder with a feature attention system for the encoded images. With those parameters, the comparable Chen et al model achieves a BLEU-3 score of 0.181 [1], while the similar recurrent model built achieves a better score of 0.27. One of the biggest reasons for this could be the fact that the Chen et al model seeks to caption all types of figures, not just line figures. The better score could also probably be explained by the subject replacement and unrolling augmentation resulting in single sentence, generalized descriptions, which are easier to generate for the decoder. When comparing with the more advanced systems of Chen et al, the BLEU-3 score of the recurrent model falls short, with the Chen et al system scoring 0.324 for BLEU-3 [1].

4.4 Potential improvements

During the end stages of the development of the model the biggest issue with the model architectures that seemed to arise was the ResNet used for encoding the images. Namely, the pre-trained ResNets used as encoders have been trained on general images of the real world that have lots shapes, varying colors and heavy contrast between the colors. This is very different from figures, where the figures are light-colored solid or dashed lines, plotted on a white background with some black x- and y-axes.

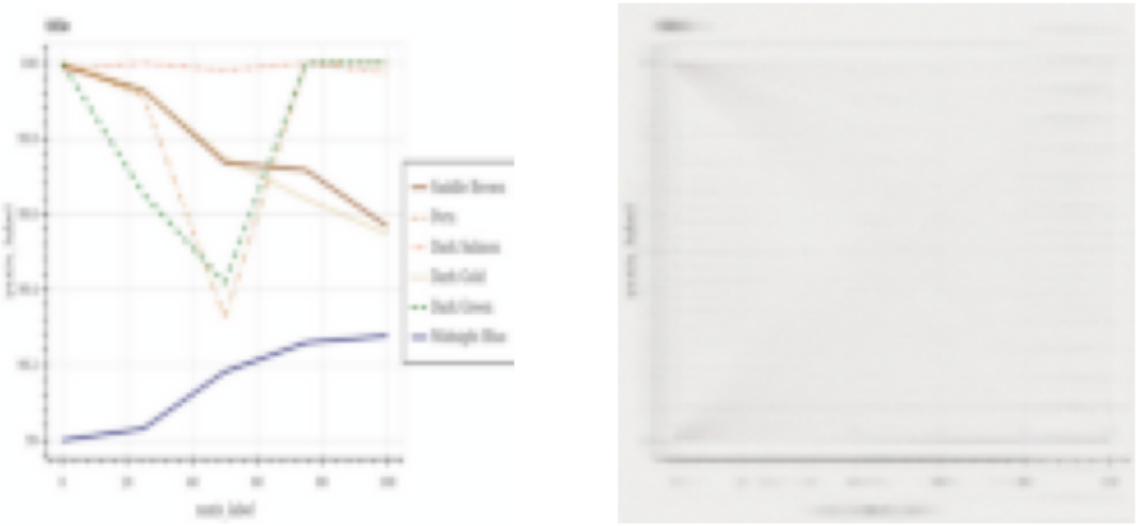

Figure 7. an example of the image inputted into the encoder on the left, and the output of the autoencoder on the right. The output of the autoencoder has defined axes, but the lines are very faint and don't match the original figure.

This was further confirmed by attempting to build an autoencoder based on the pre-trained ResNet used for the recurrent and the baseline model, with the objective of re-constructing the figure image from the output of the encoder. When visualizing the output of the autoencoder in Figure 7, axes are presented, and faint outlines of many different lines are presented, indicating that the encoder doesn't provide enough information to the autoencoder. When inspecting the output of the encoder by hand, encoded images are very similar as well, with the mean absolute difference between encoded figures being as low as 0.002.

The best solution to this problem would be to use a different encoder. An encoder built with the ResNet architecture would most likely still work, but it would have to be trained from scratch, rather than using the pretrained version of it that was trained for general purpose image recognition.

Another improvement that could be done is to generalize the captions and to introduce more metadata about the figure to the decoder. This would allow the decoder to rely on more information than is generated by the encoder, which could improve the descriptions. When adding more metadata, more attention systems for those features could help as well, as they did in Chen et al [1].

Another potential improvement would be to add high-level descriptions, similar to what is used in Chen et al with the high-level descriptions of FigCAP-H [1]. When generating high-level descriptions, it also makes sense to provide the word embeddings of subjects in the figure to the decoder. If high-level descriptions were used with the templating approach, the model would only need to learn to list the subjects along with a repetitive sentence structure around it. If subject replacement were to be enabled, the model would learn to output a static list of subject replacement tokens ("<A>","<B>","<C>"...) that would later be replaced with the actual subjects. In any case, high level descriptions prepended to the detailed descriptions could improve the perceived quality of the descriptions.

Finally, another approach to improvement would be to shift the figure captioning problem from captioning the image of the figure to captioning the data that is used to construct the figure, similar to Obeid et al [7]. This approach might offer better results, simply because it is easier for the model to caption the numerical data, rather than having to extract complex features from the image of the figure. This approach places some limits on which figures could be captioned, since the data required to construct the figures must be available. However, given the high performance and very natural output of this approach (as demonstrated in Obeid et al [7]), it seems to be a fair tradeoff.

5. Conclusion

In this thesis, image and figure captioning architectures were described, a dataset was constructed and two different figure captioning models were trained and presented. The models that were constructed are capable of describing figures with varying degrees of accuracy and correctness.

To construct the dataset, a pre-existing question-answer dataset generation software was written according to an existing approach. The dataset generation software includes some augmentations inspired by other approaches. With that dataset, two structurally different models were built, trained and evaluated on the generated dataset.

The first model, the "baseline model", uses template and machine learning based figure captioning. The model predicts a trend and two values from the figure that behave according to this trend, information that is then combined to form a description. The baseline model performs moderately well by correctly describing a good amount of figures in the test set, although with some nonsensical descriptions.

The second model, inspired by a pre-existing approach, doesn't perform as well. While it learns to generate the basic structure for the captions, it fails to accurately describe the figures.

Possible reasons for the low performance of the second model were also presented, with some future improvements, that could lead to better performance for models created with similar architectures in the future.

6. References

[1] C. Chen, R. Zhang, E. Koh, S. Kim, S. Cohen, T. Yu, R. Rossi, R. Bunescu. Figure Captioning with Reasoning and Sequence-Level Training. 2019.
https://arxiv.org/abs/1906.02850 (03.12.2020)

[2] I. Goodfellow, Y. Bengio, A. Courville. Deep Learning. MIT Press. 2016.
https://www.deeplearningbook.org/ (10.02.2021)

[3] C. Aggarwal. Neural Networks and Deep Learning. Springer. 2018.

[4] H. Sepp, J. Schmidhuber. Long Short-Term Memory. Neural Comput 1997, pp 1735–1780. doi: https://doi.org/10.1162/neco.1997.9.8.1735 (25.02.2021)

[5] O. Vinyals, A. Toshev, S. Bengio, D. Erhan. Show and Tell: A Neural Image Caption Generator. 2014.
https://arxiv.org/abs/1411.4555 (25.02.2021)

[6] C. Liu, L. Xie, Y. Han, D. Wei, X. Yuan. AutoCaption: An Approach to Generate Natural Language Description from Visualization Automatically. *2020 IEEE Pacific Visualization Symposium (PacificVis)*, Tianjin, China, 2020, pp 191-195, doi: 10.1109/PacificVis48177.2020.1043.
https://ieeexplore.ieee.org/document/9086209 (02.12.2020)

[7] J. Obeid, E. Hoque. Chart-to-Text: Generating Natural Language Descriptions for Charts by Adapting the Transformer Model. 2020.
https://arxiv.org/abs/2010.09142 (05.12.2020)

[8] S. Demir, S. Carberry, K. McCoy. Summarizing Information Graphics Textually. 2013.
https://www.aclweb.org/anthology/J12-3004/ (15.02.2021)

[9] S. Kahou, V. Michalski, A. Atkinson, A. Kadar, A. Trischler, Y. Bengio. FigureQA: an Annotated Figure Dataset For Visual Reasoning. 2018.
https://arxiv.org/abs/1710.07300 (03.12.2020)

[10] H. Kaiming, Z. Xiangyu, S. Ren, J. Sun. Deep Residual Learning for Image Recognition. 2015.
https://arxiv.org/abs/1512.03385 (15.03.2021)

[11] K. Papineni, S. Roukos, T. Ward, W. Zhu. BLEU: a Method for Automatic Evaluation of Machine Translation. 2002.
https://www.aclweb.org/anthology/P02-1040/ (03.04.2021)

Acknowledgements

Special thanks to Eduard Barbu, my supervisor, for evaluating and accepting my thesis idea and providing continuous support when working on the thesis - by evaluating ideas, suggesting relevant and related work and commenting on the writing process.

Another thanks to Margus Niitsoo, my colleague and thesis seminar group instructor. He helped me greatly by providing feedback and inspiring motivation during the latter stages of writing the thesis.

The author of this thesis receives a stipend from IT Academy (IT Akadeemia).

Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, Sander Nemvalts,

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

"Annotating Line Graphs With Machine Learning",

supervised by Eduard Barbu.

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Sander Nemvalts
06/05/2021