UNIVERSITY OF TARTU
Institute of Computer Science
Computer Science Curriculum

Kaur Karus

Artificial video generation from storytelling

Bachelor's Thesis (9 ECTS)

Supervisor(s): Assoc. Prof. Gholamreza Anbarjafari
Mr. Suman Sarkar

Tartu 2017

Artificial video generation from storytelling

Abstract:

In this Bachelor’s Thesis, the main purpose is to develop a solution for automatically creating a visual representation of contextually linked series of sentences. For this purpose, an application was developed using Python that uses pre-processed text to generate a series of images pertaining to the text. This series can then be combined into a video that illustrates the content and context of the sentences.

Keywords:

Video generation, text analysis

CERCS: P170 - Computer science, numerical analysis, systems, control

Sidusast tekstist video genereerimine

Lühikokkuvõte:

Selle bakalaureusetöö eesmärgiks on arendada programm, mis looks seostatud lausete põhjal lausete konteksti väljendava visuaalse kujutuse videona. Pythonis arendatud programm kasutab lausete eeltöötluse tulemit, et genereerida piltide seeria, mis kirjeldab lausetes toimuvat. Loodud pilte saab seejärel omavahel kokku panna, et saada tervet teksti kirjeldav video.

Võtmesõnad:

Video genereerimine, tekstianalüüs

CERCS: P170 - Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine (automaatjuhtimisteooria)

Table of Contents

1 Introduction ... 4

2 Overview ... 5

2.1 Pre-Processing ... 5

2.2 Processing ... 7

2.3 Post-Processing ... 7

3 Direct model ... 12

3.1 Description ... 12

3.2 Word information ... 12

3.3 Memory ... 13

3.4 Video creation ... 13

4 Object-oriented model ... 14

4.1 Description ... 14

4.2 Word information ... 14

4.3 Memory ... 15

4.4 Video creation ... 15

5 Results ... 17

5.1 Video ... 17

5.2 Limitations ... 24

6 Conclusions ... 25

7 References ... 26

Appendix ... 28

I. Style templates ... 28

II. License ... 29

1 Introduction

Many solutions exist to turn various types of input into text. Machine learning can be used with an accuracy that equals or exceeds human ability [1, 2, 3, 4] or describe the content of a random image [5, 6, 7]. However, few or no programs process text to create other types of media. Such a solution could be used for creating automated videos from theatrical scripts or children's books or illustrating a story being told in real time. To fulfil this task, machine learning must be used.

In general, machine learning systems have been built by finding patterns in data and running statistical analysis on those patterns. Depending on the results of the analysis, further actions are taken. Traditionally, finding these patterns has been done manually [8]. In recent years, using deep learning algorithms has become increasingly popular for solving more complex problems where patterns are difficult to define [9, 10, 11]. Deep learning algorithms simulate simple neural networks to take advantage of representational learning.

Artificial neural networks have been proven to be useful for processing information created by humans, including speech [12, 13] and handwritten text [14, 15]. To use deep learning, the neural network must be given a set of raw or modified input values and corresponding representative output values. This way, the network itself can look for patterns within the input data to accurately predict the results. During the learning process, each pattern's effect on the results is evaluated, including compound effects of multiple patterns. After learning, the network can predict outputs of previously unprocessed source information.

To use deep learning algorithms, it is necessary to have sample inputs and corresponding expected output videos. The purpose of this Thesis is to ease this process by creating a program that takes a sample text and creates a series of images that represent the context of the text. Sentences are first processed using the Stanford CoreNLP toolkit [16], which analyses the construction of each sentence and tries to find meaningful connections between words and sentences. The result is then passed on to the program that uses the information to build an internal model of the sentence context following predefined patterns and create a series of sequential images based on each sentence. The resulting images can then be combined into a video. For the purposes of the Thesis, a further step was taken to process the resulting images to stylize the video using an artificial neural network [17].

The Thesis consists of multiple parts. The second Chapter describes the process structure used for creating a representative video. The third and fourth Chapters give detailed descriptions of an initial and a final model used by the program for word information processing. The fifth Chapter describes the results of the program, followed by program limitations. The final Chapter summarizes the Thesis.

2 Overview

2.1 Pre-Processing

Creating a video based on text requires information about the context of each sentence. That means analysing the sentences to find out the relation of each word and phrase to other words and phrases in the text. For this purpose, the Stanford CoreNLP [16] toolkit was used. The CoreNLP toolkit is a set of commonly used natural language processing tools that allow to annotate each word as a token with tags describing its lemma, meaning the word's canonical form, part of speech (POS), meaning the word's syntactic category, word dependencies and location in the general text. Furthermore, it can detect words and phrases that allude to information specified in earlier sentences, hereby called coreferences. For each coreference the referencing phrase and referenced phrase are annotated with their location in the text and sentence.

The CoreNLP toolkit is ready to be run directly after download, which makes it simple to set up and use. Running the toolkit on a text in a .txt file results in an easy to read XML document that describes each word and word dependency in each sentence, followed by coreferences that connect multiple sentences. The resulting file is the primary source of information for the program created for this Thesis. The structure of the file, as depicted in Graph 1, is a tree divided into major branches. This is especially useful as each of the major branches is used in different stages of sentence processing. Each leaf-type element holds a string value, which makes processing it a programmatically straightforward task.

The secondary source of information is a database of nouns that can be depicted by an image. This database can be updated independently from the created program itself and define any number of words. Each word in the database is linked to an image that can be loaded by the final program. The database used for the Thesis was created using a collection of images publicly available for free use in combination with art created by the author of this paper. The collection was then analysed by a script that, depending on the words used in the text, attempts to find images depicting words that are present in the sentences. If the words are already present in the database, no further search is made. This approach was chosen for testing purposes where new word support can be added by adding pictures without the necessity to manually edit or recreate the database.

To decide whether an image file depicts a certain word, the file name is checked. The first image file name that contains the word, more specifically the lemma of the word, being searched for is assumed to depict the word and added to the database. An exception is made if the word is part of a longer word in the file name, such as 'sun' in 'sunflower'. To avoid compound nouns that can change the meaning of the word, file names where the word is immediately preceded or followed by alphabetic characters are ignored for the purposes of the search.

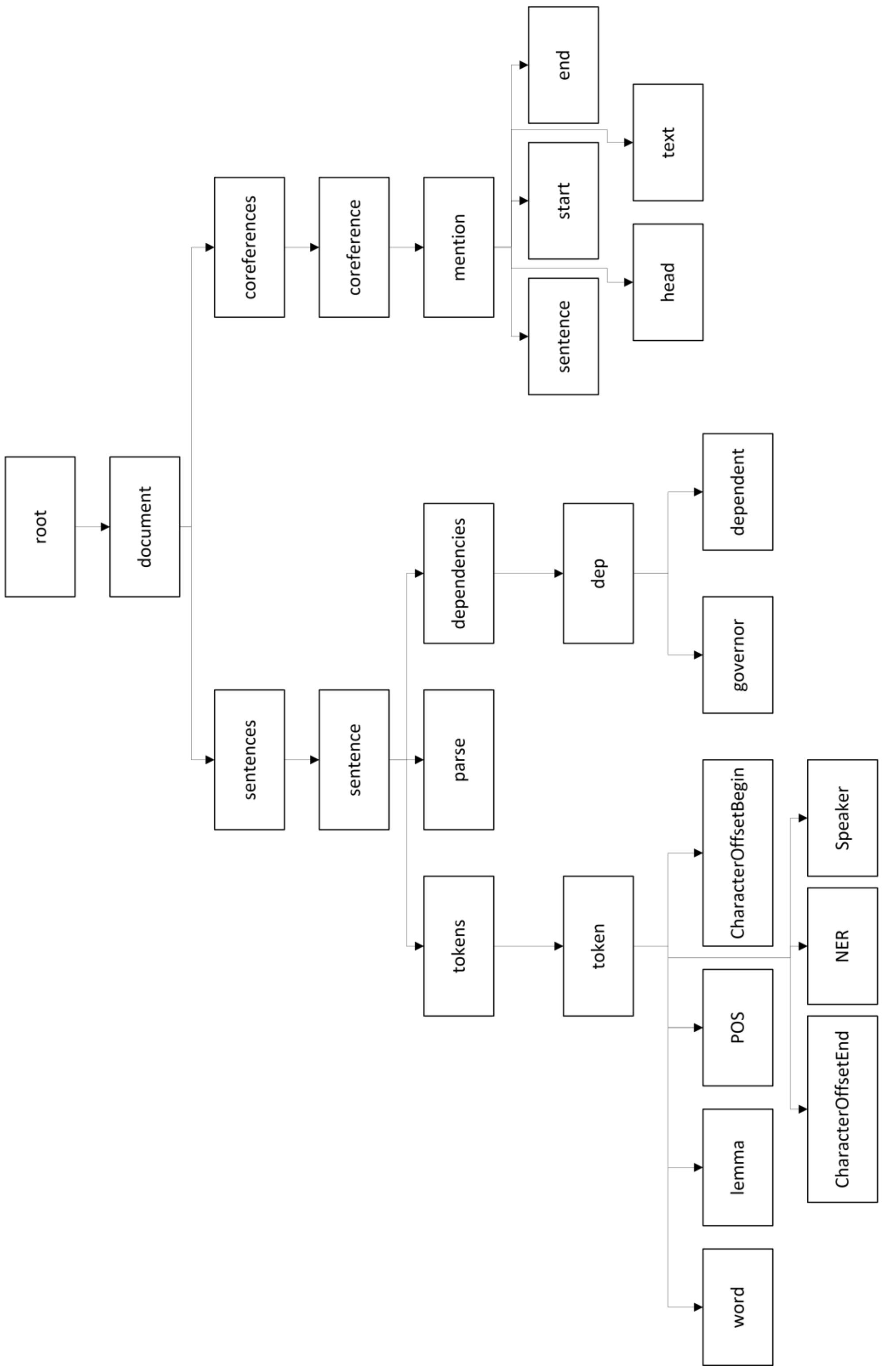


Figure 1. Structure of CoreNLP generated XML file.

2.2 Processing

Once the sentences have been analysed and a dictionary has been created, the main processing step can take place. The processing script was written in Python 3.5 using PIL library for image processing and TkInter library for creating each frame on a canvas. Furthermore, the cElementTree library was used for extracting information from the XML file created by the CoreNLP toolkit. The precise approach of processing is explained in the following two chapters.

2.3 Post-Processing

After creating all the frames, it is apparent that the images on each frame do not always complement each other in terms of the whole composition. In other words, the actors appear as disjoint parts and do not feel as part of the whole. To make each frame, and hence the entire resulting video, become artworks, it was decided to process each frame using neural style transfer.

Neural style transfer is a method by which the style of an artwork, e.g. a painting, is transferred to another using an artificial neural network. By using the same style in the entire scene, each frame becomes an individual wholesome work of art. At first, Justin Johnson's neural style transfer solution [18] was used. The solution uses the Torch package to load a Caffe neural network that is then taught a style from an image. The source style used originated from Van Gogh's "Starry Night". Using the created frame as a template, the style was added over 450 iterations, finalizing a stylised frame about every 5 hours on a HP Elitebook 840 laptop (Intel i5-5200U CPU, 8GB RAM, no dedicated GPU). The number of iterations was chosen as a compromise between effect and duration of processing. It was also apparent that too much processing would make frames too abstract to be contextually relevant.

Due to an unknown reason, Johnson's solution failed to stylise every frame, leaving a few with very little styling or completely untouched. Since the stylised frames were significantly darker and bluer, poorly stylised frames stood out as a flicker in the final video. To detect remove the problematic frames, the colour composition of each frame was analysed. A Python script that averaged the amount each RGB colour was present per pixel per stylised frame was created and the results were entered into Microsoft Office Excel.

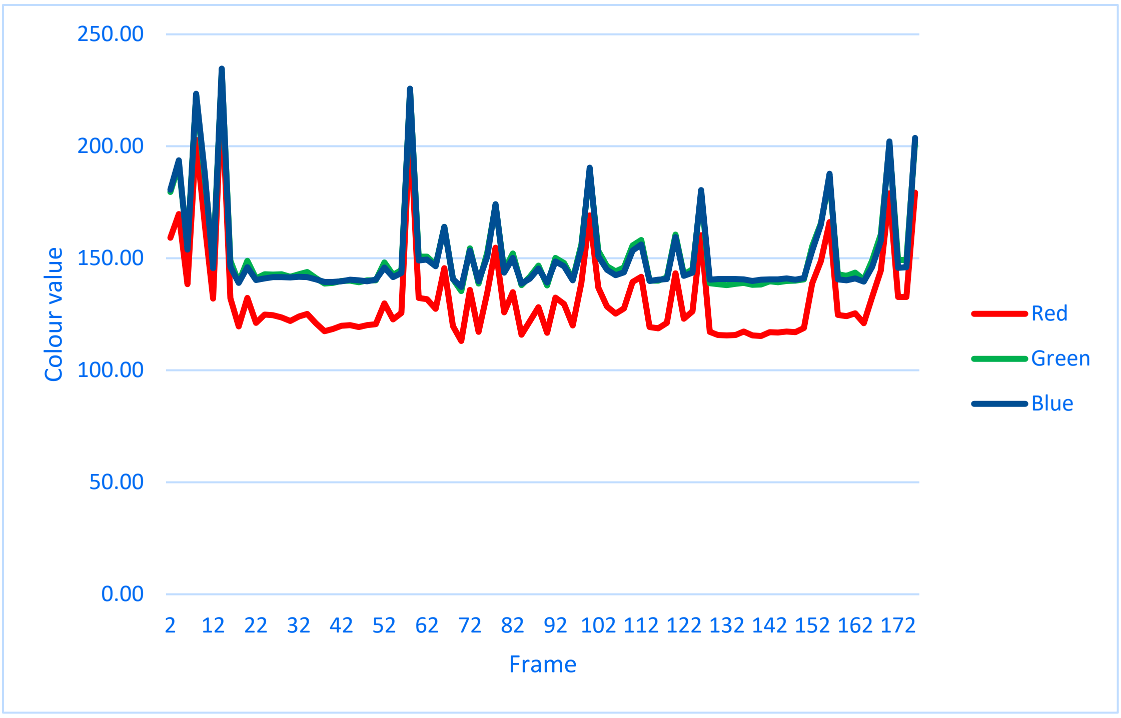


Figure 2. Average RGB values per frame.

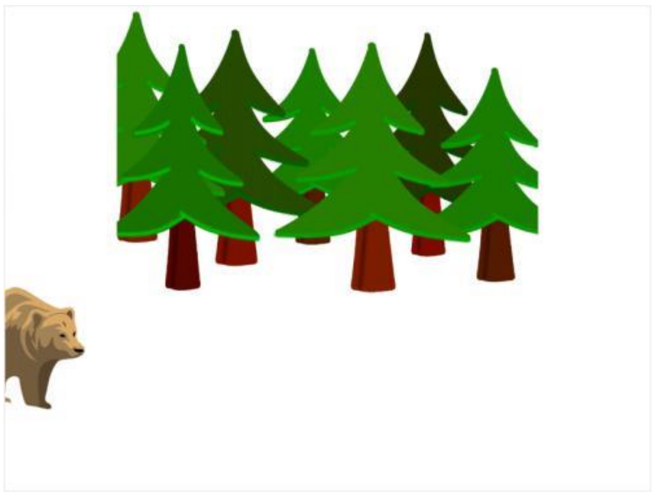
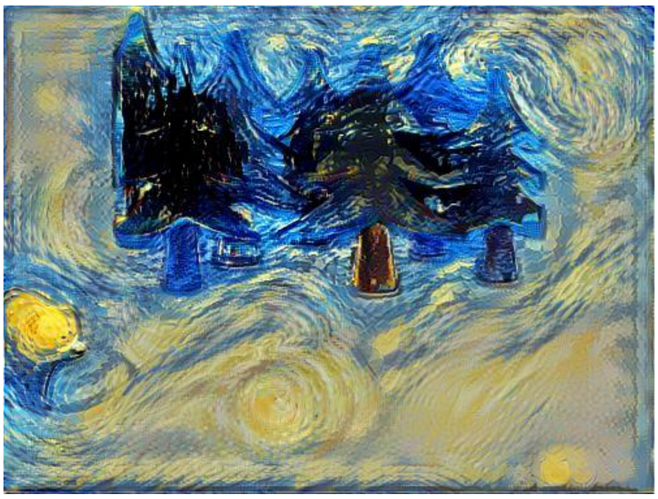
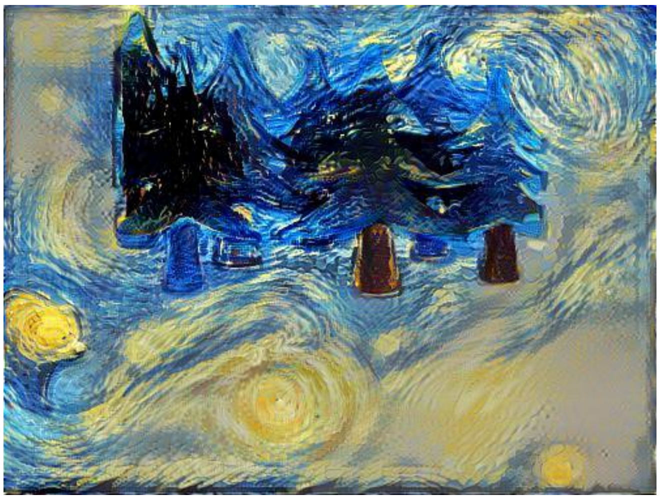
The resulting Graph 2 showed peaks in all colours in the frames that were less stylised. This was to be expected as problematic frames used mostly white as background colour while fully stylised frames used shades of blue. Using the data, a further Python script was created to create a list of all problematic frames which could then be systematically removed. The script averaged the amount of red colour on each frame on a scale of 0 to 255 and if the computed average of red on a single frame differed from the average amount of red contained per pixel per previous suitable frame, as defined in Formula 1 as $\Delta\overline{R_n}$, by more than 30 units, the frame would be classified as unsuitable for the video and its filename would be appended to a text file. Otherwise the frame's average would be added to the total amount of red observed per frame and the average would be recalculated. For analysing the first frame, a default value of 160 was used for the comparison value as most initial frames corresponded to this value. The filenames in the created file are then discarded from the list of frames used to create the video. This approach significantly decreased flickering, but caused actors to move with inconsistent speed.

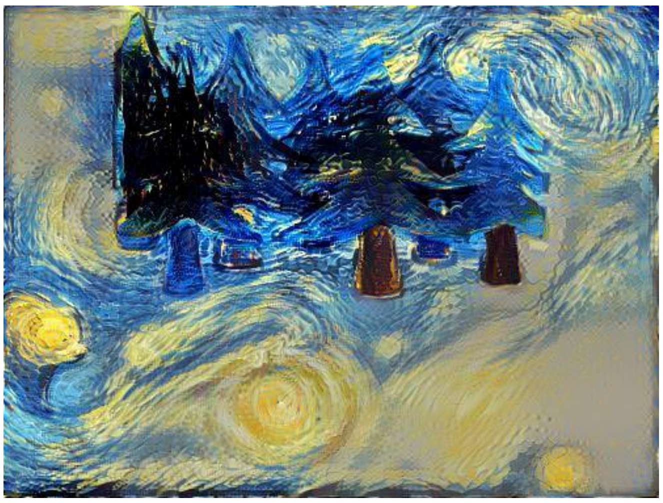
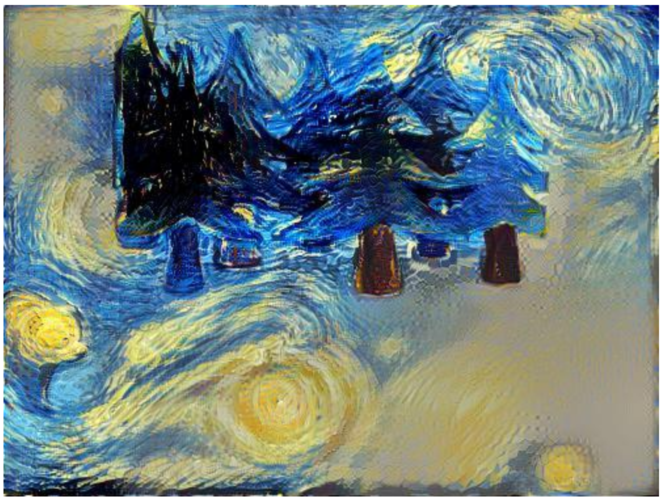
$$\Delta\overline{R_n} = \overline{R_n} - \left(\sum_{i=1}^{n-1} \overline{R_\iota}\right) \div (n-1) \tag{1}$$

A significantly better result was achieved using Dmitry Ulyanov's solution [19], which is based on Johnson's work. Initially, the style template originated from "The Starry Night", but further trials with Claude-Auguste Renoir's "Banks of the River" produced slightly better results. A sample of both paintings is located in Appendix I. The newly chosen solution reliably stylised all frames at a slightly faster rate using the same setup. It processed each frame over 160 consecutive iterations, finishing a frame every 80 minutes. The reduced number of iteration was chosen due to the new solution having a greater effect on a frame per iteration than the previous solution. As seen in Table 1 demonstrating the effect the number of iterations of stylisation run on a frame, no significant changes occur in the final frame after 160 iterations. The change from 80 to 160 iterations, however, is notable by the increase in detail of the forest. Beyond 160 iterations, some elements, e.g. the bear, gradually become less recognisable.

All suitable stylised frames were compiled into a list in a text file. All images on the list were combined into a single video using the FFmpeg [20] framework. For the purposes of the Thesis, videos were encoded with the libx264 video codec at 12 frames per second.

Table 1. Stylisation effect using Ulyanov's solution.

Number of Iterations	**Result**
0	
80	
160	

Number of Iterations	Results
240	
320	
400	

3 Direct model

3.1 Description

The direct model uses a series of groupings of words. Each grouping includes words with a similar observed quality. The quality in question can be word class, e.g. ‘noun’, or how the word relates to the overall scene, specifically the location of an object described by a noun in the set scene. When it comes to creating each frame, each word is looked for from each grouping for directions on its intended appearance and location or lack thereof.

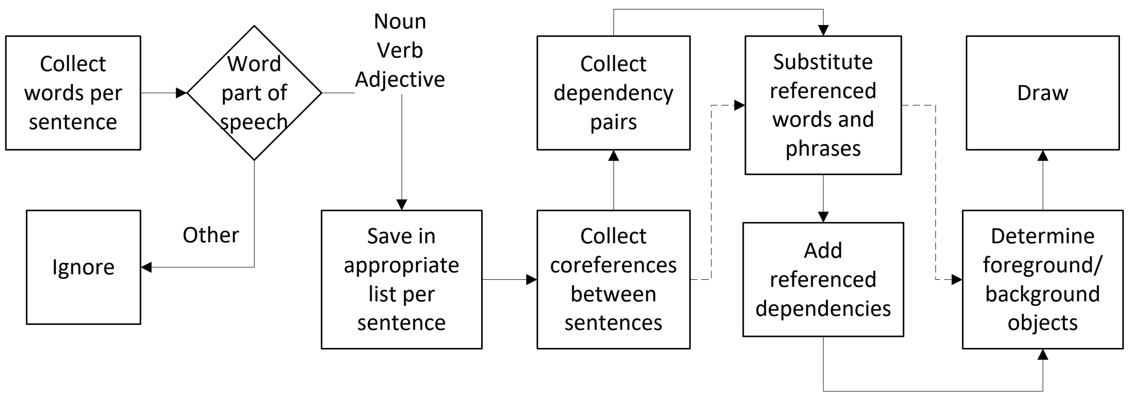

Collect words per sentence
Word part of speech
Noun
Verb
Adjective
Other
Ignore
Save in appropriate list per sentence
Collect coreferences between sentences
Collect dependency pairs
Substitute referenced words and phrases
Add referenced dependencies
Determine foreground/ background objects
Draw

Figure 3. Simplified process flow of the direct model.

3.2 Word information

The first major task was to read relevant information from the output file created with CoreNLP toolkit. Based on information read from the file, all words in each sentence are divided into three main groups: adjectives, nouns and verbs. These three groups were chosen to be significant as they contain information that can, in general, be easily visualized. Other parts of speech, such as adverbs and articles, are ignored.

The toolkit is also able to find previous references of words and phrases. This allows the program to understand that words with a definite article, such as ‘the camel’, and prepositions, such as ‘it’, are words and phrases denoting something that were first introduced in a specific previous sentence. Since the purpose of the program is to draw videos based on a series of connected sentences, knowing how the sentences relate to one another is paramount, and information concerning previous references must be preserved. In the direct model, all referencing phrases in the sentences were directly replaced by the phrases being referenced for all later purposes. In addition, any dependencies between the words in the original referenced phrase are added to the dependencies list of the referencing sentence. These dependencies are later used to determine whether a word describes the background or foreground.

In order to understand how words within a sentence relate to each other, word dependencies must be analysed. This information can then be used to classify nouns as words describing the background or foreground objects. In a simplistic view, if a noun is dependent on another noun, the former is probably part of the background. For example, in a phrase ’a camel in a desert’, a ’camel-desert’ nominal modifier type dependency exists, where ’camel’ is the governor and ’desert’ is the dependent. It also defines how nouns relate to the verbs, for example whether a word is denoting an actor or a subject. This is crucial in longer sentences that can contain multiple verbs relating to different actors. However, since the direct model does not support linking words to each other, the only dependencies observed were of noun-noun type and are used to determine words describing elements of the background. This limitation was the main motivator for developing the object-oriented model.

3.3 Memory

For the resulting video to better describe a story, the solution must partially remember past frames. Having a limited memory of past sentences means that each sentence is not a video in and of itself, but can contain parts of previous sentences. As an example where memory is important we can use the following series of sentences:

"A boy is looking at a girl.", "The girl leaves.", "The boy turns around".

Without memory, the boy would disappear or leave from the scene after the first sentence has been played out and returns for the last sentence. In such case, the implicitly intended situation would be that the boy remains in the scene, even though his actions are unspecified. Memory remedies this issue by keeping an actor present in the scene for several sentences past their last mentioning. For this purpose, a position queue logic was created – the characters who are present in the frame, but not explicitly mentioned in the sentence being analysed, are moved towards the right side of the frame. New characters enter from the left, present characters that are mentioned again moved left. If a character is omitted for three successive sentences, the rightmost character is removed from the frame. This approach resulted in excessive actor movement for each sentence and was replaced in the object-oriented model.

3.4 Video creation

Each noun that exists in the predefined dictionary is linked to a file system path to an image depicting it. If a particular noun occurs in a sentence being processed, the corresponding image is loaded from the path. All images are resized using the PIL library, taking into account that the background objects must result in larger images, foreground objects in smaller images.

For each frame of the final video, each image is added to a blank canvas. The location of each image depends on multiple variables – foreground images can be set to three different possible positions in view and two positions beyond the right and left edges out of view. The location of a foreground image on its x axis is then defined by the frame number (base 25), the position it is leaving, the position it is moving towards, its y axis defined by a preset height in combination with a small random variance.

Background images are fixed at a single location at the centre of the scene in each frame. This means that all background images are static. Multiple background images can be loaded, generally occurring when a sentence's background is expanded upon by a subsequent sentence. Instead of replacing old backgrounds with new, the images are blended into a single background. All original backgrounds appear as semi-transparent layers of the background. The object-oriented model uses a similar, but further developed solution.

4 Object-oriented model

4.1 Description

As opposed to the direct model, where words were divided into several groupings, the object-oriented model defines a `Word` class that holds all information about a single word. This information includes the word lemma, word type (noun, verb, etc.) and dependencies of the word. Nouns also include an image depicting the meaning of the word, present and target positions in scene, movement types (walking, jumping, etc.). This approach requires proper dependency linking between words, allowing adjectives, verbs and other words to have significant meaning for the video. By using dependency linking, each sentence and phrase can be represented by a single root word, a verb, while other words are directly or indirectly dependent on that word. In the process of creating a video frame, the sentence can then be traversed in a reverse post-order walk as a word tree starting from the root word.

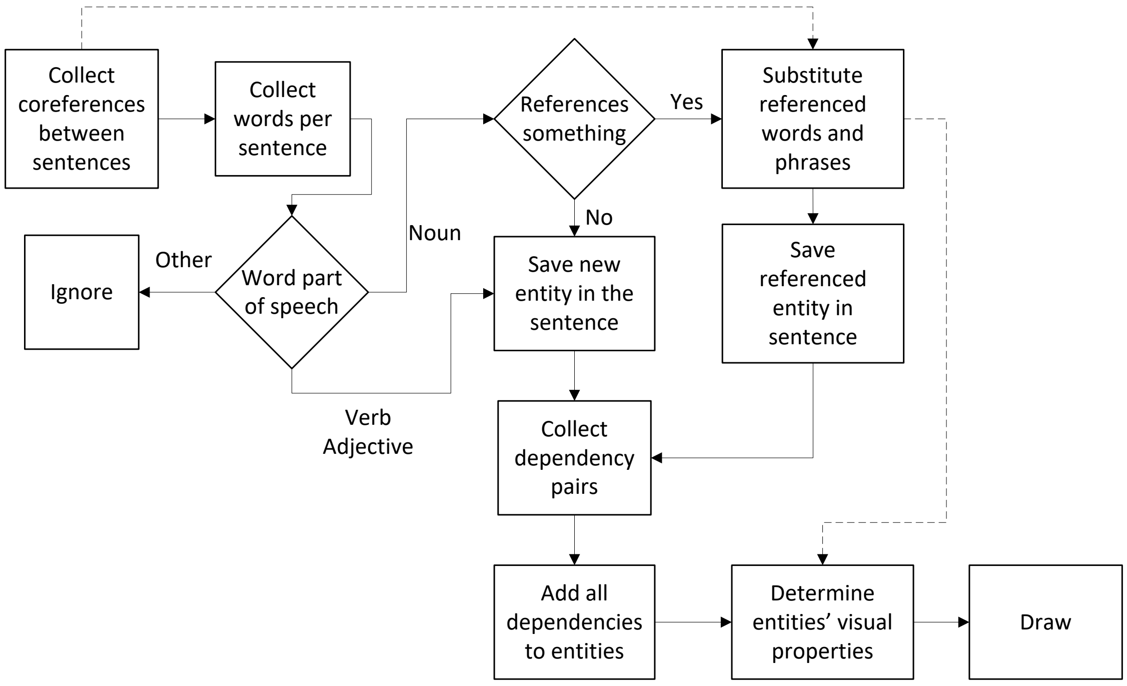

Collect coreferences between sentences
Collect words per sentence
References something
Yes
Substitute referenced words and phrases
No
Noun
Other
Ignore
Word part of speech
Save new entity in the sentence
Save referenced entity in sentence
Verb
Adjective
Collect dependency pairs
Add all dependencies to entities
Determine entities' visual properties
Draw

Figure 4. Simplified process flow of the object-oriented model.

4.2 Word information

The first step in turning a series of sentences to a series of word trees for storytelling is to find all coreferences in the text. Each coreference is simplified as a four-element array, holding the phrase referencing, the sentence number where the referencing text is located, the phrase being referenced and the sentence where the referenced original text is located. The array is then stored in another array.

The next step is processing each word separately. For each sentence, a temporary array is created for storing each word as a variable of the `word` class in the order in which they appear in the original sentence. For each word a variable is created, and if a noun being analysed appears in a coreference in the phrase being referenced, the phrase being referenced is substituted by the variable representing the noun. Referencing phrases are all in the form of a noun, preceded by 'the'. Knowing this, if a word is preceded by the definite article, a coreference is searched for. If the word is referencing a previously processed phrase, the variable representing the referenced phrase is added to the temporary sentence array, and no variable is created for the referencing word or phrase. This approach was adopted to allow a sentence to contain several actors of the same type. For instance, in a sentence with two bears, where one of them has been described earlier, the approach keeps the pre-described bear as a previously known variable, and creates a new variable for the new bear. This way the program cannot confuse the bears or mistakenly display only one of them. In addition, this approach also allows for an actor to be mentioned several times in a sentence without causing duplicate images in the final video.

Due to an aesthetic concern, a dictionary of exclusion words was created. The dictionary consists of words representing multiple similar objects, and the objects being represented. If the representative word, such as 'city', is present in a sentence, it is replaced by a variable representing information about the word. If a represented word, such as 'building', is present, it is replaced by the representing variable if possible. This process logic is alike to finding and substituting coreferences, and was implemented to avoid convoluted backgrounds where different layers consist partially or completely of the same content.

Once the temporary array is created, the next step is to add dependency connections between the newly created variables. Each dependency described by the CoreNLP toolkit specifies the dependency type, as well as the index of the governing and dependent word in the sentence, which corresponds to the respective indexes in the array of variables. Depending on the type of dependency, such as 'nominal subject', the variable representing the dependent word is linked to the variable representing the governing word. In the 'nominal subject' case, the dependent is added as a subject to a verb, and the noun receives a special tag describing the action it should take. While this approach significantly expands the variety of dependencies analysed, some dependencies, for example in the case of articles 'a' or 'the' being dependent on their nominal heads, are dismissed without further action.

Backgrounds are also determined through dependencies. In this model the 'case' dependency type, which is used in conjunction with prepositions such as 'in', 'under', 'by', is used to define words denoting objects in the background. Whether a variable depicts something in the background or the foreground is stored in a Boolean flag within the variable.

4.3 Memory

The final step before drawing any frames is checking for actors who are not explicitly present in the new sentence, but were in previous sentences. For this purpose, a collection of all mentioned words was created. Words in the collection that are not present in the sentence queued to be drawn are added to the frame first. This causes actors in the new sentence to be drawn over all other present actors, bringing the most relevant information – the content of the newest sentence – forward. Every time a depictable noun is found in a new sentence, it is moved to the end of the drawing queue to be drawn last.

Instead of forgetting any actors after a pre-set number of subsequent sentences where the presence of actor is not explicitly stated, as used in the direct model, no words are ever forgotten. However, actors are permitted to leave the scene upon special verbs, such as 'leave' and 'depart', by being assigned to an off-scene position. Any off-scene actors are not displayed.

4.4 Video creation

As in the direct model, each noun can have an image linked to it. In the object-oriented model, each variable of the `Word` class can have an image as an instance variable. This means that the image does not have to be reloaded every time a noun is processed, which includes words collected in memory, as it would in the direct model. The images are stored after resizing appropriately. Due to a limitation of the dictionary, further explained in Chapter 6, white backgrounds are removed from images for a more aesthetically pleasing result.

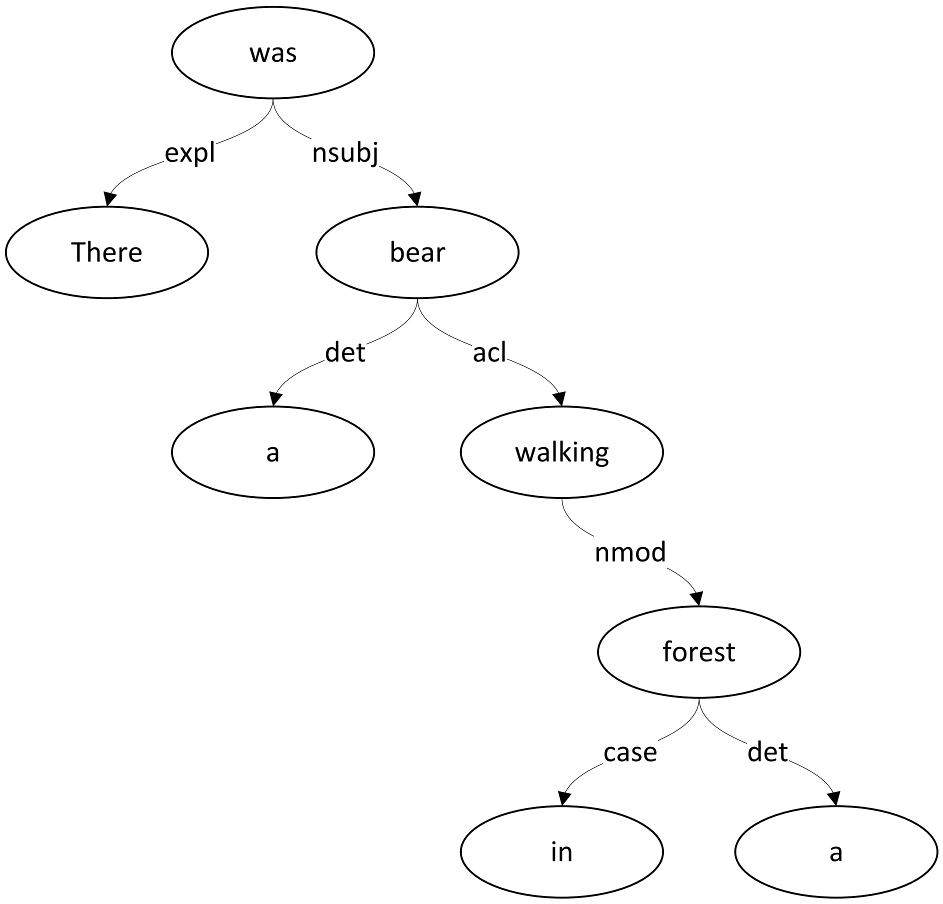


Figure 5. Sentence structure tree.

To begin drawing, all information is loaded for a single sentence. Then the dependency links between words are added, creating the word tree as described in Graph 4. The root of the tree is always a verb. This is followed by a check to collect actors and other information present in previous sentences, but not stated in the sentence being processed.

For each frame, each word in the word tree and in the memory defines its precise location or effect in the frame, dependent on information given to them. To assist in choosing location, an array was created that counts the number of actors in or moving to any given possible foreground position. While multiple actors can generally not occupy the same in-scene position, some verbs, such as ‘pounce’ can apply an exception.

Once each image location is fixed for a frame, the images are added to a blank canvas. As described in Chapter 4.3, the words that were previously depicted, but not mentioned in the sentence being processed, are drawn first. This approach dictates that in the case of multiple backgrounds, the image in the topmost layer originates from the last sentence specifying a background. As the same is true for actors, the resulting frame places the active participants of the most recent sentence to the front while leaving bystanders remain aside or behind them.

5 Results

5.1 Video

For some scenarios, the program gives produces a very relevant video. The quality of the video is strongly dependent on the used model and stylisation solution. The resulting frames from the direct model are shown in Table 2. For a better overview, only every eighth frame is demonstrated (frames 1, 8, 16, 24, 32, 40, 48, 56). The same frames are demonstrated using Justin Johnson's neural art solution, as described in Chapter 2.3, in Table 3. The sample scenario for the first model was:

There is a cat in the desert.

The cat was walking under the sun.

The cat found the bird.

The cat and the bird found a dolphin.

The cat and the dolphin conversed.

The dolphin was swimming under the sun.

The cat stood.

A total of 25 frames are created for each sentence, which is apparent in the video – while the cat is depicted in the desert as described in the first sentence (frames 1-25), the sun is not displayed until the second sentence (frames 26-50), the bird only enters at the third sentence (frames 51-75). The continues presence of the desert demonstrates that "the cat" in the sentences is detected to be referencing "a cat in the desert". The lack of the sun in the third sentence is indicative of the lack of memory in the first model where backgrounds are concerned. The memory implemented for foreground actors causes them to reposition themselves in the order they are mentioned, starting from the fifth sentence. The last image (frame 56) in the stylised video is, in fact, the 57th frame created due to poor stylisation in frame 56 leading it to be omitted as part of the deflickering process. The stylisation in each frame is visibly different in each frame, most notably in the direction of created patterns on the originally white area and the darkness of the cat. The visible differences in patterns and brightness in the final stylised frames causes the resulting video to appear subjectively more chaotic than intended.

Tables 4, 5 and 6 show results (frames 1, 8, 16, 24, 32, 40, 48, 56) created with the object-oriented model without any stylisation and with Ulyanov's stylisation solution, also described in Chapter 2.3, using two different paintings as source style. The scenario for the object-oriented model was chosen to be more complex:

There was a bear walking in a forest.

It saw a fox under a tree.

The fox had been talking to a rabbit, and did not notice the bear.

The fox attacked the rabbit.

The bear attacked the fox.

The fox was pinned down against the ground, so the rabbit ran away.

The bear was victorious.

As with the direct model, 25 frames were created for each sentence. In this scenario, the most common referencing phrase is “the bear”, which always references “a bear walking in a forest”. Unlike the direct model, the object-oriented model does not strictly require the forest background to be referenced to remain in the scene. This avoids situations similar to the one present in the previous example where the sun was omitted from the video after it was no longer mentioned within the sentences. On the other hand, this means all possible backgrounds are superimposed on each other, which can cause problems in other scenarios. All objects within the scene are stored in memory, giving no cause for foreground actors to rearrange their positions. However, in Table 4 it can be noted that some verbs have a visible effect on the position of actors within frames. This can be seen in the sentence “It saw a fox under a tree” whereby the fox appears in the scene from the right side. This is due to the definition of “see”, which causes the object and subject, if not already present in the scene, to approach from opposing sides. This follows the logic that it is more common for one to see objects ahead, not behind, of oneself. The default actor approach direction is from the left. The approach movement tempo was also altered for the characters to arrive in the scene faster and waiting in their positions for the following sentence, giving the viewer a chance to focus better on the new actors.

The frames depicted in Table 5 show a more uniform art style more befitting a video than the previous solution. However, the movement and presence of a large yellow circle similar to the one visible in the original painting by van Gogh, lead to the change in painting used to stylize. The second painting for style transfer was Pierre-Auguste Renoir’s “Banks of the River”, consisting of more green tones, which is more befitting for a forest setting. A sample of both paintings is in Appendix I. As seen in Table 6, using the new painting as a style template results in even more uniform frames that result in a more natural impression of the entire video.

Table 2. Results with direct model without stylisation.

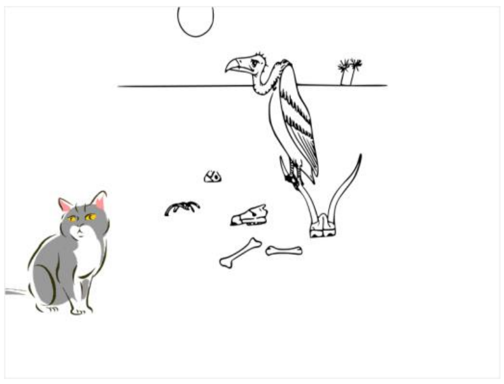
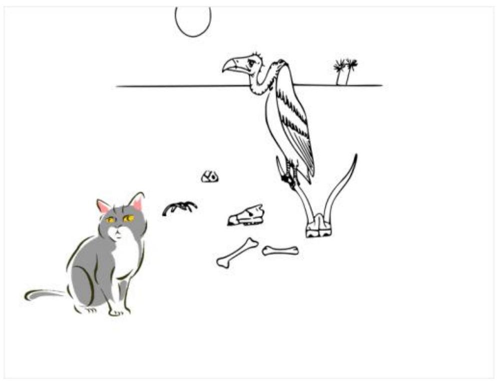
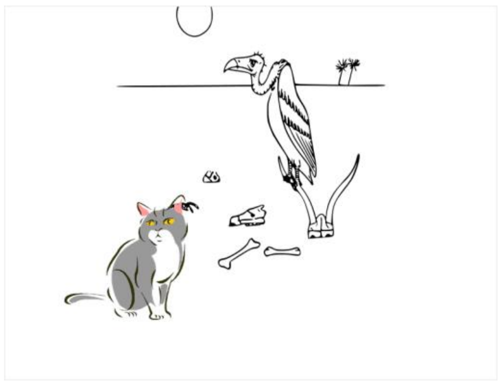
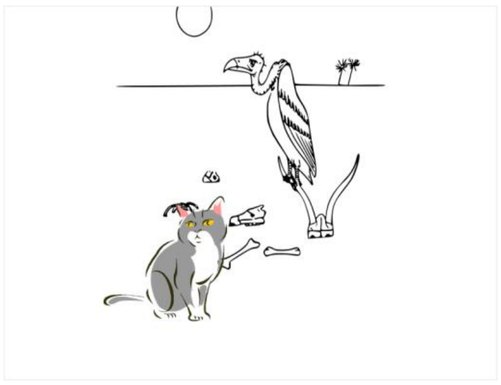
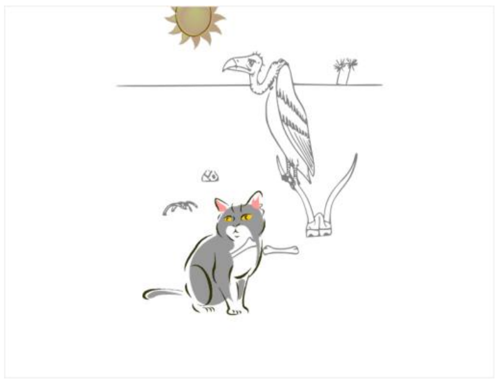
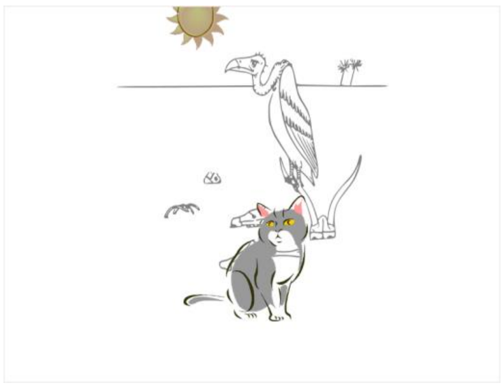
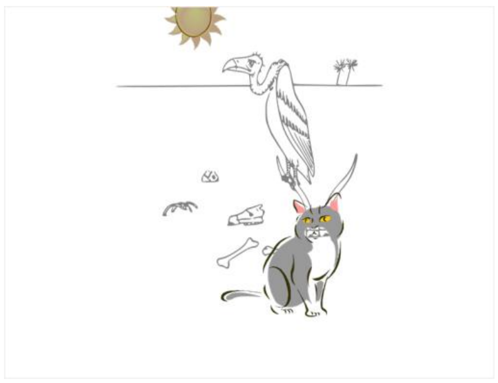
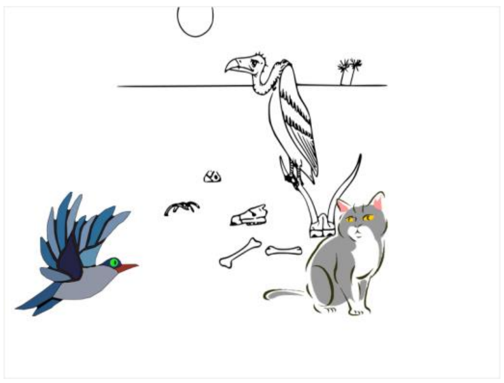
	
	
	
	

Table 3. Results with direct model with Johnson's stylisation.

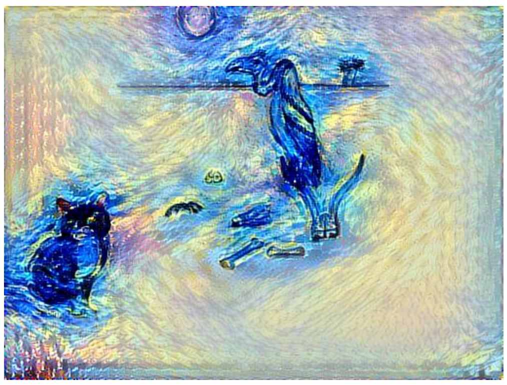
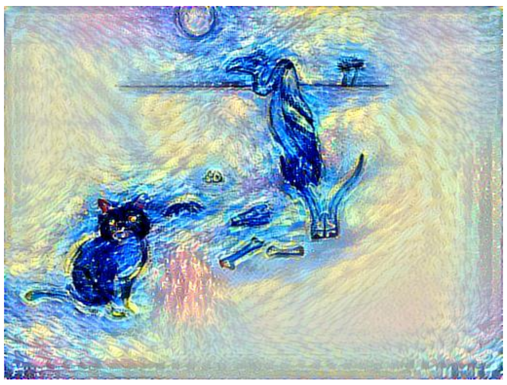
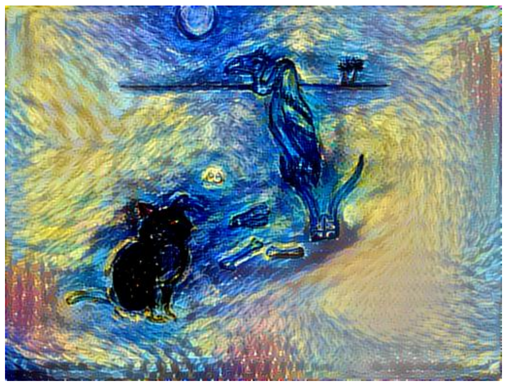
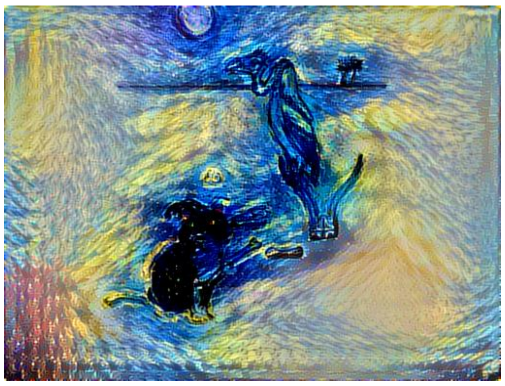
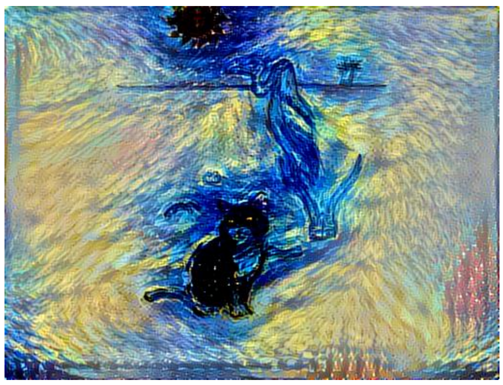
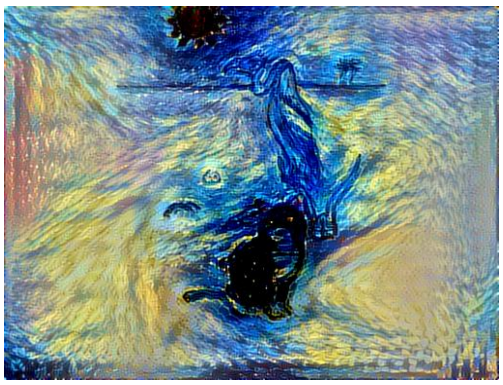
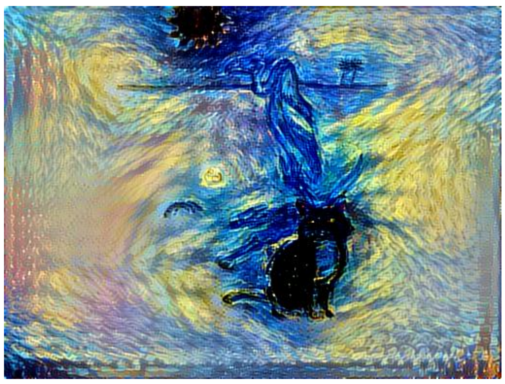
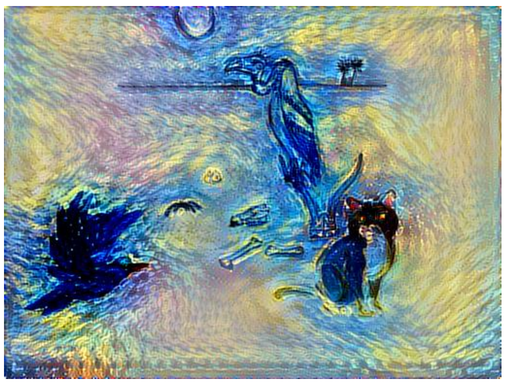
	
	
	
	

Table 4. Results with object-oriented model without stylisation.

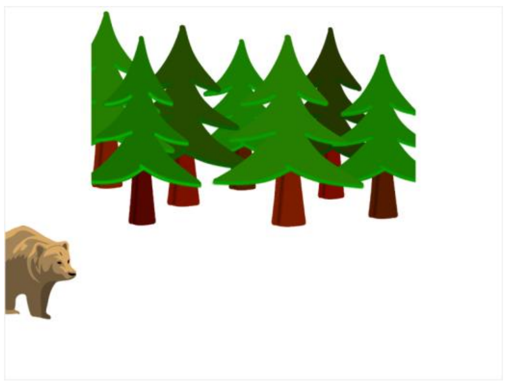
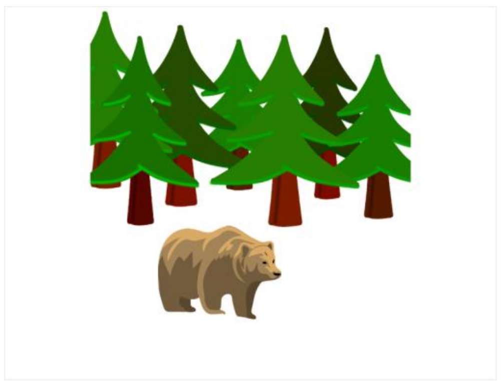
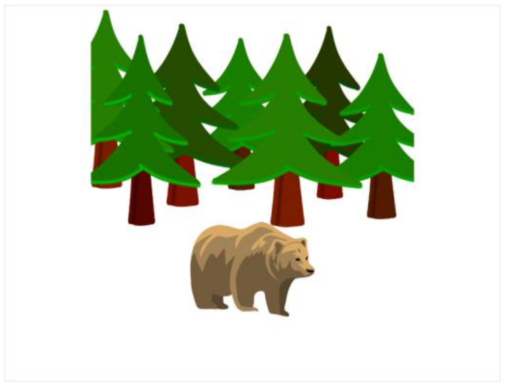
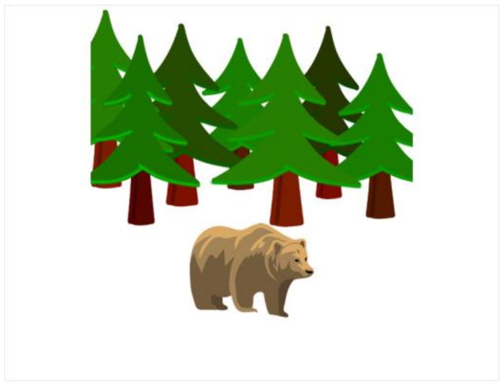
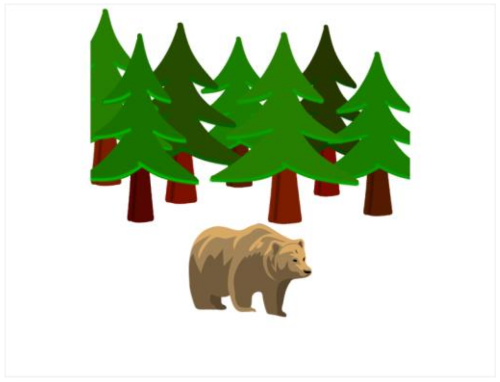
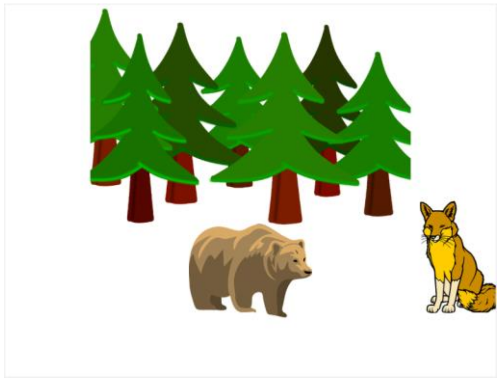
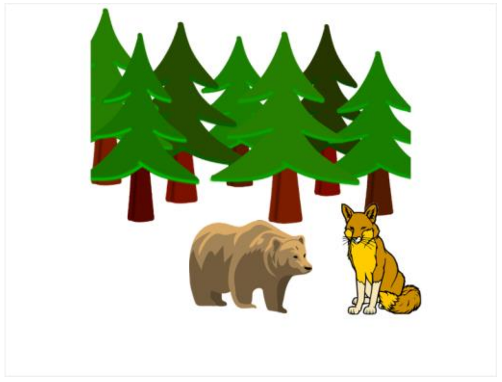
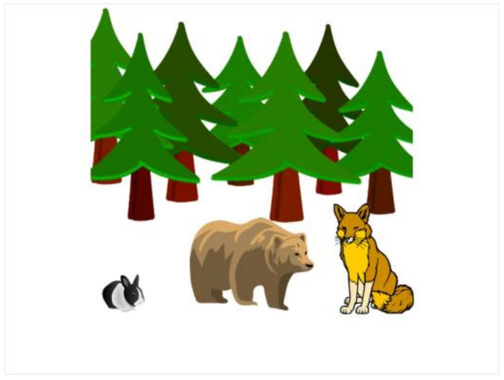
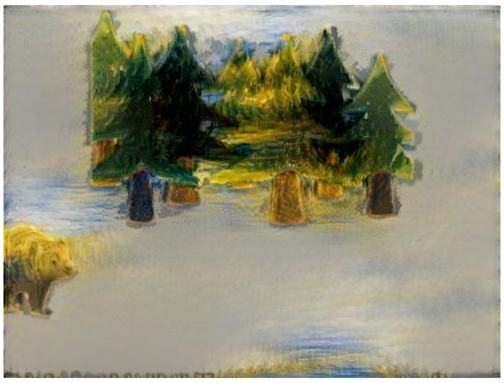
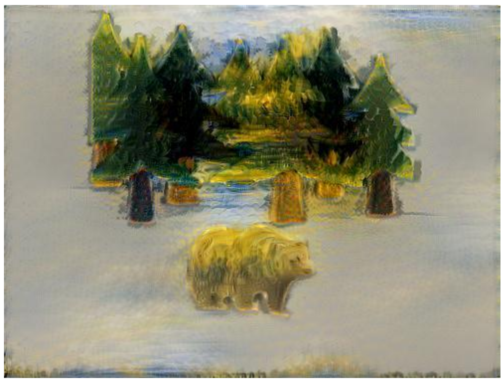
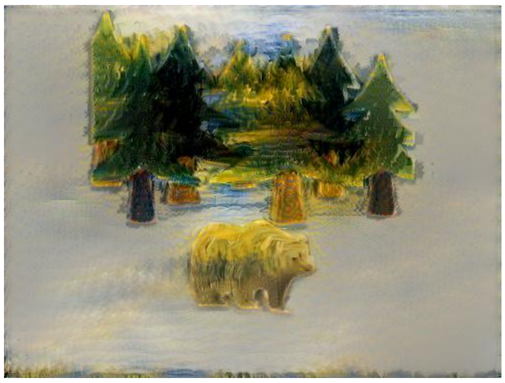
	
	
	
	

Table 5. Results with object-oriented model with Ulyanov's stylisation (van Gogh).

Table 6. Results with object-oriented model with Ulyanov's stylisation (Renoir).

5.2 Limitations

Two major limitations were found during the development. Firstly, the images present in the dictionary cannot be assumed to contain a transparent background. Images with a white background cause sharp white edges to cover up other images in the frame. This was remedied by defining white as a transparent colour for each loaded image. However, this caused some images, such as an image depicting a cat with white hair, to become transparent in the middle. A more complex solution would have to be implemented to ensure the transparency filter only affects the outermost white area.

Secondly, the abilities of the program are greatly limited by the dictionary. Words that are not defined in the dictionary cannot be depicted as images. As a corollary, the range of stories that can be turned into video form is highly restricted by the size and composition of the dictionary.

The size of the dictionary does not only limit the variety of depictable nouns, it also limits the variety of verbs that can be depicted. In the present solution, each defined verb alters the movement trajectory of an actor by defining a new formula for calculating the coordinates in each new frame. However, this requires many verbs to be defined within the program code, instead of a dictionary file. While a single complicated formula with multiple variables that each verb defines can be used to simplify defining new verbs, no formula can describe all possible trajectories. This is exacerbated by the unknown number of dependents of each verb – "Boy looks at a bear. They hug." and "A boy hugs a bear." differ significantly from each other from a linguistic and a programmatic standpoint while both use the same verb "hug". A proper definition would have to be general enough to handle any number of objects and subjects, yet any verb without a definition cannot be visualized.

The same is true for adjectives. It is impossible to correctly visualize "big", "yellow" or "striped" without defining these words beforehand. Since adjectives can affect almost any property of a depictable noun, no general solution exists for defining adjectives. For the purposes of the Thesis, the author was limited to defining only various nouns and verbs.

6 Conclusions

In the process of the Thesis, a solution was developed for creating a representative video from contextually linked sentences – a story. This task was inspired by the abundance of solutions solving the task in the opposite direction – visual material turned into text – with few or no solutions creating visual representations from text and the number of potential use cases ranging from visualizing theatrical scripts to real-time story illustration. The solution can be used for creating example data for representational learning. More specifically, the data can be used to teach an artificial neural network to solve the same task in a more generalized manner, allowing it to be used in a larger variety of situations.

To complete this task, two fundamentally different models were explored, of which the more flexible on was further developed. Both models relied on plaintext, processed with the CoreNLP toolkit. After creating individual frames for the finalized video, each frame was stylized using an artificial neural network. Once each frame was post-processed in a suitable manner, all frames were combined into a single video representing the entire story.

The final solution is mostly limited by its dictionary, mostly being able to illustrate the meaning of words and phrases that have been explicitly defined for the purposes of the program either within the code itself or in a separate dictionary file following a predefined definition pattern. The larger the dictionary, the more variability the scenarios that can be reliably illustrated can have.

7 References

[1] W. Xiong, J. Droppo, X. Huang, F. Seide, M. Seltzer, A. Stolcke, D. Yu ja G. Zweig, „Achieving Human Parity in Conversational Speech Recognition,“ Microsoft Research, 2017.

[2] T. Joachims, „Text Categorization with support vector machines: Learning with many relevant features,“ *Machine learning: ECML-98*, 1998.

[3] M. Libbrecht ja W. Noble, „Machine learning applications in genetics and genomics,“ *Nature Reviews Genetics,* nr 16, pp. 321-332, 2015.

[4] E. Mayfield ja C. Rosé, „Open Source Machine Learning for Text,“ *Handbook of automated essay evaluation: Current applications and new directions*, 2013.

[5] T.-H. Huang, A. Agrawal ja P. Kohli, „Visual Storytelling,“ *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2016.

[6] R. Klanten, S. Ehmann ja F. Schulze, Visual Storytelling: Inspiring a New Visual Language, Gestalten, 2011.

[7] A. Woodside, S. Sood ja K. Munz, „Creating and interpreting visual storytelling art in extending thematic apperception tests and Jung's method of interpreting dreams.,“ *Luxury Fashion and Culture,* Emerald Group Publishing Limited, 2013, pp. 15-45.

[8] Y. Lecun, Y. Bengio ja G. Hinton, „Deep learning,“ *Nature,* kd. 521, pp. 436-444, 2015.

[9] Y. Bengio, I. Goodfellow ja A. Courville, „Deep Learning,“ *Nature,* nr 521, pp. 436-444, 2015.

[10] Y. Bengio, „Learning deep architectures for AI,“ *Foundations and trends® in Machine Learning,* nr 2, pp. 1-127, 2009.

[11] L. Deng ja D. Yu, „Deep learning: methods and applications,“ *Foundations and trends® in Machine Learning,* nr 7, pp. 197-387, 2014.

[12] R. J. Weiss, J. Chorowski, N. Jaitly, Y. Wu ja Z. Chen, „Sequence-to-Sequence Models Can Directly Transcribe Foreign Speech,“ *Interspeech 2017,* Stockholm, 2017 (pending publication).

[13] F. Noroozi, M. Marjanovic, A. Njegus and S. A. G. Escalera, "Fusion of classifier predictions for audio-visual emotion recognition," in *23rd International Conference on Pattern Recognition (ICPR). IEEE,* Cancun, 2016.

[14] K. Gregor, I. Danihelka, A. Graves, D. J. Rezende ja D. Wierstra, „DRAW: A Recurrent Neural Network For Image Generation,“ *32nd International Conference on Machine Learning,* Lille, 2015.

[15] D. Cireşan, U. Meier, L. Gambardella ja J. Schmidhuber, „Deep, big, simple neural nets for handwritten digit recognition,“ *Neural computation,* nr 22, pp. 3297-3220, 2010.

[16] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard and D. McClosky, "The Stanford CoreNLP Natural Language Processing Toolkit," [Online]. Available: http://www.anthology.aclweb.org/P/P14/P14-5010.pdf. [Accessed 15 February 2017].

[17] A. J. Champandard, "Semantic Style Transfer and Turning Two-Bit Doodles into Fine Artwork," in *nucl.ai Conference 2016,* Vienna, 2016.

[18] J. Johnson, *neural-style*, GitHub, 2015.

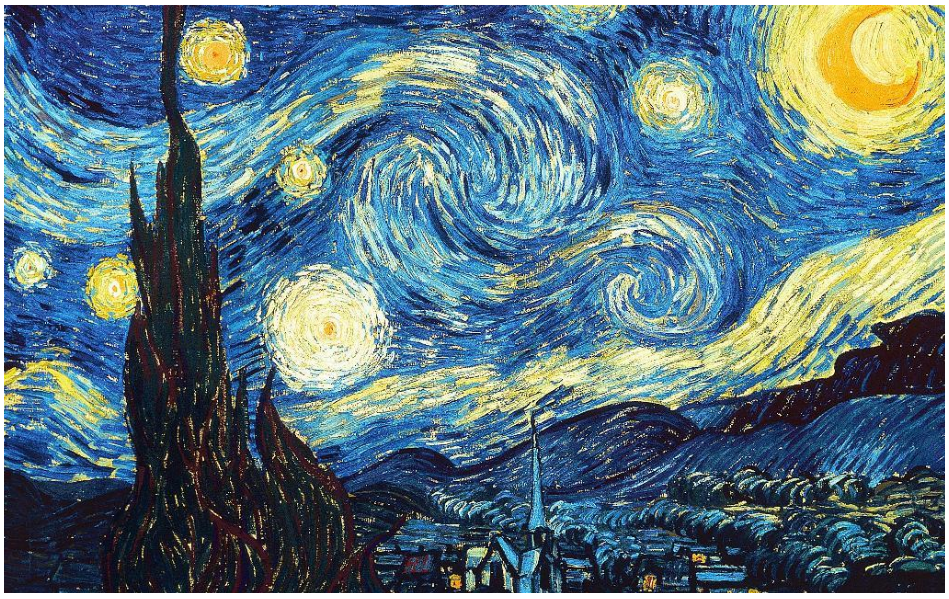
[19] D. Ulyanov, „Faster neural doodle," 2016.

[20] "FFmpeg," [Online]. Available: https://ffmpeg.org/.

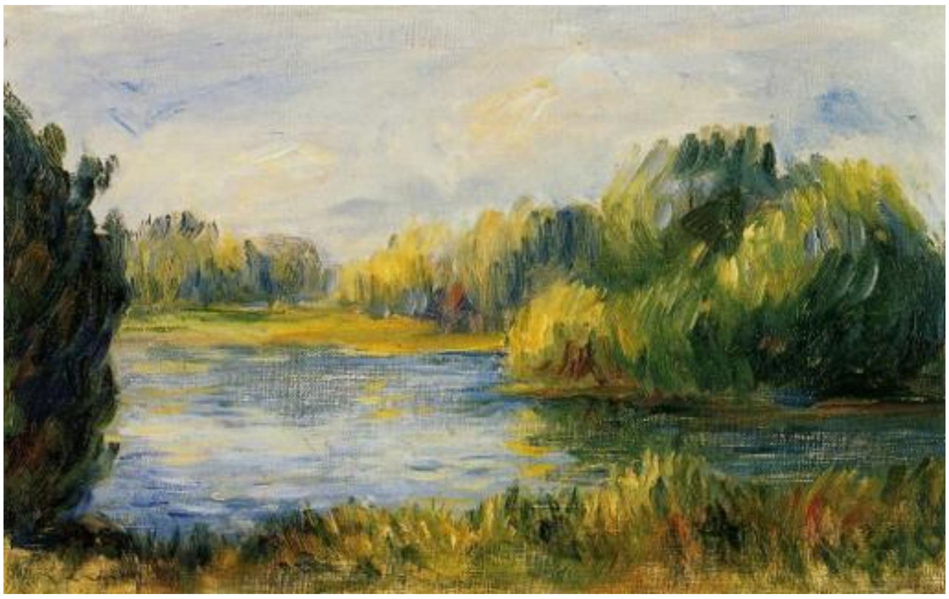
[21] K. Narasimhan, T. Kulkami and R. Barzilay, "Language Understanding for Text-based Games Using Deep Reinforcement Learning," 11 September 2015. [Online]. Available: https://arxiv.org/abs/1506.08941. [Accessed 1 May 2017].

Appendix

I. Style templates



Vincent van Gogh „Starry Night“



Pierre-Auguste Renoir „Banks of the River“

II. License