RTL Nederland: Captioning News Footage

Participants: Daan Odijk (company representative), Maurits van der Goes (company representative), Pascal Mettes (academic leader), Thomas Mensink (academic leader), Arthur Câmara, Yunlu Chen, Shuo Chen, Jiaojiao Zhao, William Thong, Emiel van Miltenburg, Devanshu Arya, Sarah Ibrahimi

Context and problem

The RTL Nieuws team produces hours of video content every day. A typical eight o'clock broadcast contains around ten edited clips and with hourly broadcasts and online shorts this adds up to more than 30 edited clips every day. These are each compiled from multiple sources for raw footage, both from newswires and from our own reporters.

All this raw footage and edited clips are manually archived for future reuse. RTL Nieuws' documentation department spends over 50 hours every week on adding annotations and metadata to all this footage. On the other end, editors search the archive for relevant footage around current news events.

We believe that with the right ICT solution we can make this a less labor-intensive process and potentially even increase the retrievability of the content. For this, we need a deep understanding of the video content and how to automatically generate (textual) metadata from this. We are looking for experts in computer vision and natural language generation to join our team.

In a first pilot project, we have found that current cloud solutions such as Google Cloud Video Intelligence[1] and Microsoft Video Indexer[2] can be leveraged to provide additional metadata tagging based on audiovisual content. However, these do not reach the level of visual detail nor the emotional depth that our archivists deliver and that our editors are using to find the context.

We are seeking to close this gap in vocabulary between the searching editors and the automatically generated metadata. We need models that understand content on a deeper semantic level, including emotional expressions and visual language such as the mise-en-scène. Another part of the solution might be found in automatically generated video captions. As part of the ACM MM Challenge in 2016 and 2017, Microsoft Research has run the Video to Language challenge[3] on their Video to Text (MSR-VTT) dataset, a large-scale video benchmark for bridging video and language. The dataset contains about 50 hours and 260K clip-sentence pairs in total, covering the most comprehensive categories and diverse visual content, and representing the largest dataset in terms of sentence and vocabulary. Given an input video clip, the goal is to automatically generate a complete and natural sentence to describe video content, ideally encapsulating its most informative dynamics.

[1] https://cloud.google.com/video-intelligence/
[2] https://vi.microsoft.com/
[3] http://ms-multimedia-challenge.com/

Research approach

In the Figure below, we show the rough overall pipeline of our approach. The rest of the document details the three main parts of the pipeline (the orange boxes) in the Figure.

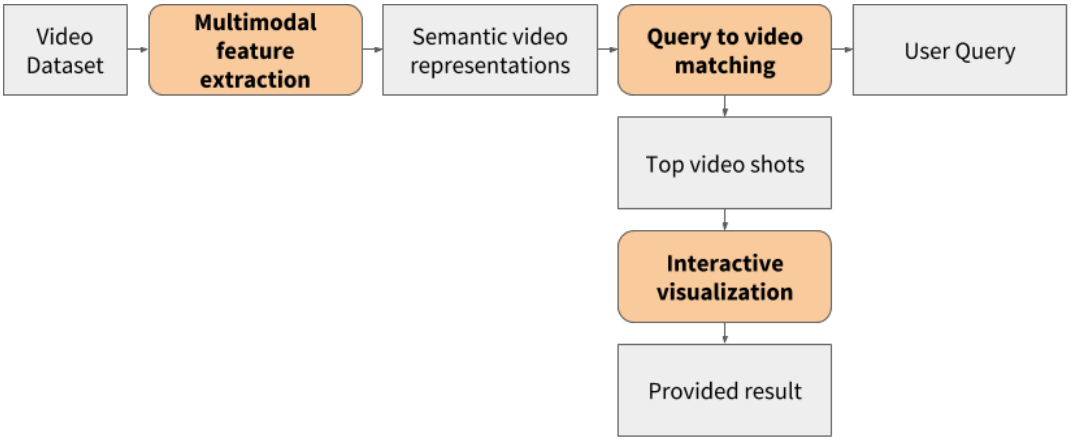

Figure 1: Overview pipeline of our multimodal feature extraction and semantic query matching for automatic video shot retrieval from text.

Part 1: Video semantic extraction

For this problem, no training set exists with mappings from videos to the desired semantics. As such, we rely on deep neural networks and on optical character recognition to provide detections from multiple types. In total, we have investigated four types: scenes, objects, people, and on-screen characters.

Scene recognition

To perform scene recognition, we employ a ResNet architecture pre-trained on the Places365 dataset. For a single frame, we output the following results:

- Classification as an indoor or outdoor scene.
- Probability distribution over 365 scene categories.
- The probability distribution over 102 scene attributes.

The results are computed every 10th frame of each video. Then, the probabilities are averaged for the frames in each shot to obtain shot-level scene predictions. The code is based on the implementation as provided on https://github.com/CSAILVision/places365 and script run_placesCNN_unified.py.

Fine object recognition

For object recognition, we follow a similar strategy and employ a pre-trained deep network and apply the network to each tenth video frame. Here, we use a ResNet-101 architecture pre-trained on 1,000 ImageNet categories. This results in a 1,000-dimensional probability distribution that is averaged over the frames in each shot. The code is based on a MxNet code implementation developed internally by the University of Amsterdam and is currently under submission as a journal paper.

Coarse object detection

Where the scores from ImageNet provide detailed object information, we have also performed within-video data extraction in the form of object detection. We perform detection every tenth frame with a Mask R-CNN network pre-trained on the MS-COCO dataset. This detection handles coarse objects such as objects, persons, cars, etc. The original pre-trained model is located https://github.com/facebookresearch/maskrcnn-benchmark.

Optical character recognition

For OCR, we employ the open-source OCR engine Tesseract. For a video, we generate character recognition using the following command:

```
!tesseract --tessdata-dir . -l nld -c tessedit_create_tsv=1 "{root + video + '/' + frame}" "{out + video + '/' + frame}"
```

After obtaining the raw OCR output we perform several forms of clean-up and shot aggregation. For frame-level OCR detections, we first perform thresholding based on the size of the detection and the confidence score. This removes false positive detections. Next, we remove punctuation marks and perform a fuzzy matching to a large dictionary. The dictionary is a combination of the words in FastText and OpenTaal. When removing punctuation marks, some words might be split into multiple words. For each case, we match both the concatenation and the separate words to the dictionary and select the one with the highest word likelihood. For aggregation to shot-level OCR, we apply both a set operation (which results in unique words in the shot) and a list operation (which results in all words in the shot).

This source code for this command can be found on the following Github page: https://github.com/tesseract-ocr/tesseract.

Action recognition

Where the object, scene, and ocr detections are performed in individual frames, we have also investigated the spatio-temporal nature of videos for action recognition. For this part, we have investigated the I3D network architecture pre-trained on the Kinetics-400 dataset. This results in a score for 400 action categories in each video shot with at least 64 frames. The code is based on the implementation as provided on https://github.com/piergiaj/pytorch-i3d.

Video file format

We aggregate the multimodal video semantic extraction into a single json file per video. The json files contain video-level and shot-level entities and scores for the different extractions. The json files contain

more meta-data about the videos themselves, but they are not used in this project, hence ignored in this documentation.

Part 2: Query matching

Given the multimodal semantic extraction for the videos in the database, the next question aims at matching the video semantics to the query words. We focus on two types of matching. The first type focuses on matching with automatic OCR detections. The second type focuses on matching with the semantic video detections.

For both parts, we rely on the Elasticsearch search engine[4] for indexing and searching over the video files described above. The standard Elasticsearch matching algorithm is BM25, a significant improvement over the classical TF-IDF with cosine similarity method.

The BM25 (or textual) search matches every textual field in the video file, attributing it a score. The score for each field (OCR, title, and description) is raised by a factor on the user-defined score, given by the slider on the user interface. The final score is given by the sum of the individual scores.

For the embedding (or semantical) search, we rely on the gensin[5] implementation of the FastText[6] word embedding algorithm. By retrieving the word embeddings from the Dutch textual representation of each of the concepts from our video models, (like "vuur"), we are able to perform an visually semantic search from the query embedding and the top-k concepts for each field. This search focuses on the visual semantics, namely: ('places', 'places_attributes', 'places_environment', 'coco', 'coco_count', 'imagenet', 'kinetics')
We use the cosine distance between the query and textual concept embeddings as a score function.

Part 3: User interaction

With a ranked match between user queries and videos or video shots, the final question is aimed at how to communicate the discovered results with the user and how to enable a first form of user interaction. To that end, we have made a front-end system where users can input queries. Based on the queries, we provide a ranking of videos with a content-based analysis. The ranking can in turn be improved with a slides-based approach. For the top detections of each part, the user can specify its importance. This enables an interactive exploration of the dataset.

[4] https://www.elastic.co/
[5] https://radimrehurek.com/gensim/
[6] https://fasttext.cc/

Results

Overall, we have managed to extract the following content-based semantics from videos:

- Extraction of on-screen information through OCR.
- Extraction of coarse objects from local detection networks.
- Extraction of fine objects from networks pre-trained on large scale image datasets.
- Extraction of global scenes from networks pre-trained on large scale image datasets.
- Extraction of actions from networks pre-trained on large scale video datasets.

Based on this extraction, we have made a matching system based on elastic search (search from the text itself) and based on word embeddings.

To combine both parts and enable a user interaction, we have developed a front-end visualization with the option for interaction. Below, we show a screenshot of our user interaction front-end with an example query. The sliders enable a fast exploration of the video content depending on the type of result the user is looking for.

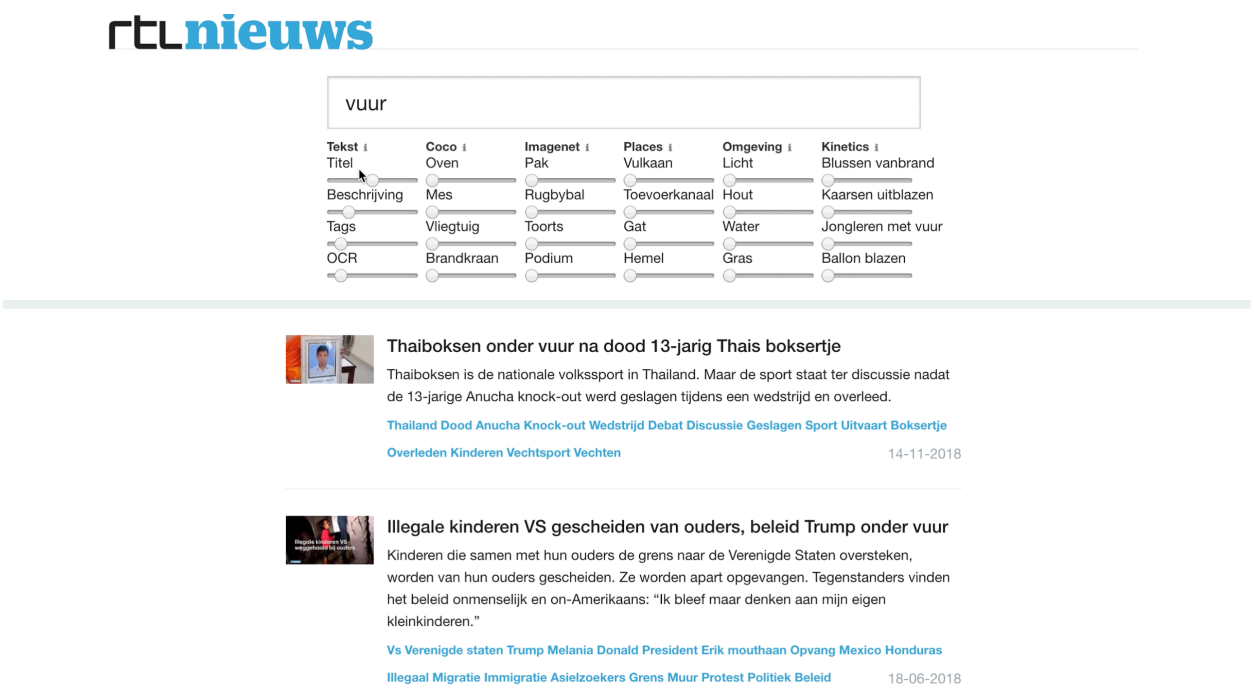

Figure 2: Visualization of the interactive front-end to search through the video data.

Future work

The proposed pipeline provides a full pipeline from query to video for retrieval. It is however not complete and can benefit from improvements in several aspects:

- Currently, a set of 490 videos is used as database. A natural next step is to expand this set to investigate how well the proposed approach works in a large-scale setting.
- Our system now enables search by Dutch keywords. Extensions include queries from other languages and queries from descriptions, rather than keywords.
- While qualitatively, we have achieved interesting results, an overall measure of performance is currently lacking. Future works can be to generate a test set with known queries and matches to directly measure the performance of the system or a user study to detail how effective our overall system is for recall, precision, and exploration.