Dublin City University and Partners' Participation in the INS and VTT Tracks at TRECVid 2016

Mark Marsden[1], Eva Mohedano[1], Kevin McGuinness[1], Andrea Calafell[3], Xavier Giró-i-Nieto[3] Noel E. O'Connor[1], Jiang Zhou[1], Lucas Azevedo[2], Tobias Daudert[2], Brian Davis[2], Manuela Hürlimann[2], Haithem Afli[4], Jinhua Du[4], Debasis Ganguly[4], Wei Li[4], Andy Way[4], Alan F. Smeaton[1]*.

[1]Insight Centre for Data Analytics, Dublin City University, Dublin 9, Ireland
[2]Insight Centre for Data Analytics, National University of Ireland, Galway, Ireland
[3]Universitat Politècnica de Catalunya, Barcelona, Spain
[4]Adapt Centre for Digital Content Technology, Dublin City University, Dublin 9, Ireland

Abstract

Dublin City University participated with a consortium of colleagues from NUI Galway and Universitat Politècnica de Catalunya in two tasks in TRECVid 2016, Instance Search (INS) and Video to Text (VTT). For the INS task we developed a framework consisting of face detection and representation and place detection and representation, with a user annotation of top-ranked videos. For the VTT task we ran 1,000 concept detectors from the VGG-16 deep CNN on 10 keyframes per video and submitted 4 runs for caption re-ranking, based on BM25, Fusion, word2vec and a fusion of baseline BM25 and word2vec. With the same pre-processing for caption generation we used an open source image-to-caption CNN-RNN toolkit NeuralTalk2 to generate a caption for each keyframe and combine them.

1 Introduction

A team of researchers from Dublin City University (Insight and ADAPT Research Centres), National University of Ireland, Galway (Insight Research Centre) and Universitat Politècnica de Catalunya, took part in the TRECVid 2016 annual benchmarking [1], which is part of the TRECVid series [17] which first started in 2001. The team completed runs for two tasks, namely Instance Search (INS) and the new showcase pilot task on Video to Text Description (VTT), both the matching and caption generation sub-tasks. These are described in this paper.

*Contact author: alan.smeaton@dcu.ie

2 Instance Search (INS)

This year's Instance Search task consisted of finding people at specific locations. We proposed a baseline system based on pre-trained Convolutional Neural Networks (CNN). We used VGG16-faces [10] and VGG16-Places205 [22] to extract face and place image representations.
We submitted four runs with the following official results:

	Type	**Id submission**	**mAP**	**P5**	**P100**
Run 1	Automatic	I_A_insightdcu_4	0.031	0.593	0.133
Run 2	Automatic	I_A_insightdcu_3	0.059	0.780	0.243
Run 3	Interactive	I_E_insightdcu_2	0.031	0.527	0.139
Run 4	Interactive	I_E_insightdcu_1	0.036	0.780	0.165

Table 1: Runs with associated *mean Average Precision* (mAP) and *Precision at k* (P@K)

A brief description for each run follows:
- Run 1: Automatic baseline based in VGG16-faces + BoW VGG16-places205
- Run 2: Automatic baseline (Run 1) augmenting the query face with the top-20 faces obtained.
- Run 3: Interactive run using user annotations of results obtained in Run 1. Final rank consisted of re-sorting final ranks by fixing positive annotation on top and discarding negative annotations.
- Run 4: Interactive run using user annotations of results obtained in Run 2. Final ranks were constructed as in Run 3.

2.1 Dataset

Participants in the TRECVid Instance Search were provided with 244 video files (300GB, 464h) with associated metadata, each containing a week's worth of BBC EastEnders programs in MPEG-4/H.264 format. Additionally, each video is divided into shots of short duration (between 5 seconds and 2 minutes).

People and locations query set. The query set consists of 30 topics, each including: (1) a *person* textual description from among a total of 7 persons. (2) A *location* textual description from among a total of 10 locations. (3) 4 image examples of the *person*. (4) Between 6 and 12 image examples of the *location*.[1] (5) For each of the 4 person image queries, a *binary mask* is also provided. (6) The shot where the keyframes belongs.

Target database. In order to handle the large amount of video information provided, a target database was built by uniformly extracting keyframes from every shot with a sample rate of 1 fps. The resulting dataset contained 1,569,502 keyframes (with resolution 768x576), 471,526 shots and had a total size of 120GB.

2.2 Framework

Two independent systems based on pre-trained CNN models were used to extract representations for faces (Section 2.2.1) and places (Section 2.2.2). As illustrated in Figure 1, two ranking lists per topic were generated, one for candidate faces and one for candidate locations. The ranking lists contain the final similarity scores, generated by computing the Cosine similarity $s \subset [-1, 1]$ between l2-normalized vectors. Before fusing scores, both lists (for faces and places) were filtered

[1]Both the image examples for person and locations are keyframes from the first video out of the 224 videos provided by TRECVid.

to return a unique score by shot, which is the maximum score of all their frames. Final ranking lists were generated by taking the minimum value of place and face scores per shot.

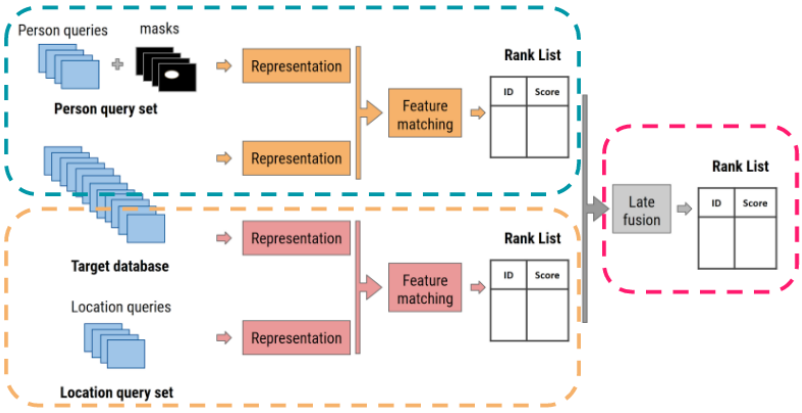

Figure 1: Framework for DCU TRECVid Instance Search 2016

2.2.1 Face detection and representation

We used *dlib* face detector integrated in the *Menpo*[2] software library to obtain a total of 964,263 faces in the target dataset. Since *dlib* misses some faces, we used the query mask to detect faces on the query face image set. For that we divide the mask into three parts (Figure 2) and generate a bounding box around the top part for the final detection in the query image.

Figure 2: Face detection in query images

For feature representation, all bounding box detections were increased by 10%, re-sized to 224x224 (no face alignment is applied) and then input into the VGG16-faces [10] network from where we extract fc7 layer activations. This layer generates a 4096-D feature vector that is l2-normalized. We index all faces taking advantage of the sparsity of the vectors. Only around 12% of 4096-D were non-zero elements. This allows us to use a CUDA-based sparse matrix library[3] to perform inverted index lookups via sparse matrix vector products on an NVIDIA GTX Titan X GPU.

For each query person, we obtained 4 face representations (corresponding to the 4 examples provided for the query face set). These descriptors were averaged and used to generate the final face ranking for Run1. Query expansion is performed by averaging the face descriptors of the top-20 retrieved faces to generate more robust face representations (Run2). With this strategy we improved our final mAP from 0.031 to 0.059.

[2] `https://github.com/menpo/menpodetect`
[3] cuSPARSE: `https://developer.nvidia.com/cusparse`

2.2.2 Place representation

We use a Bags of Local Convolutional Features approach [9] using VGG16 architecture pre-trained on Places205 [22]. Local descriptors were extracted from conv5_1 layer with the purpose of preserving the fine-grained details of the image. For that, fully connected layers were removed and images were resized to 288x336 resolution while keeping the original aspect ratio. The model generates feature maps of size (18x21) that were spatially interpolated to (36x42). With this procedure, we obtain a total of 1,512 local features of 512-D per image. Local features were l2-normalized and PCA-whitening is applied reducing dimensions to 256-D. After that, a second l2-normalization is applied. Local features were aggregated into a BoW vector of 25k centroids.

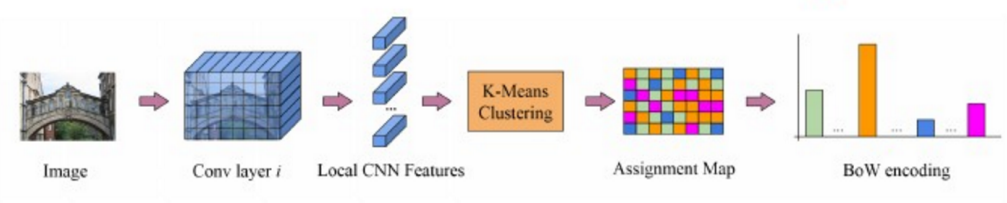

Figure 3: Bag of Local Convolutional Features Pipeline

Figure 3 summarises the CNN-BoW encoding, where the *Assignment Maps* refer to the spatial location of each visual word within the image. The 25k dimensional visual vocabulary is learned from a random subset of 3M local CNN features by using an Approximate Nearest Neighbour k-means algorithm from the *VLfeat* library [19]. Similarly, the PCA model is also learned in a random subset of 3M features using *scikit-learn* [11] Python implementation. As described for the face representations (Subsection 2.2.1, descriptors obtained for places are also sparse so, again, it is possible to benefit from the sparse CUDA-based sparse matrix for fast lookup of the queries.

For place query representation, we aggregate into a single BoW vector all visual words extracted from all keyframes provided as examples for each location. Rankings generated for locations were fixed for all runs submitted.

2.3 User Interface

Annotations generated by a user were used to re-sort final ranks obtained in Run 1 and Run 2. The front-end is based on the one created by the Insight Centre for Data Analytics the TRECVid Instance Search 2014 [7]. The interface was modified to be able to quickly annotate a large amount of keyframes and was programmed using HTML5 and AngularJS. The back-end was also modified in order to adapt it to the new annotation requirements for the TRECVid Instance Search task. As explained in Section 2.2, this benchmark requires us to create a system by fusing two approaches, one for persons and another for locations meaning three possible approaches should be evaluated separately: one for persons, one for locations and one assessing both persons and locations.

2.4 Evaluation and Normalization strategies

Despite no groundtruth data being provided, we can evaluate system performance by annotating the top-K shots in the rankings. For evaluation we use *Precision at position* K ($P@K$) for each topic, as defined in Equation (1), where M is the set of relevant images in the dataset for the query topic, and N_K is the set of the top-K elements in the ranked list under evaluation. The mean P(K) across the whole set of query topics Q defines the *Mean Precision at* K ($mP(K)$).

$$P@K = \frac{|M \cap N_K|}{K} \quad (1)$$

in order to evaluate the fusion strategies, the first step is to make sure that the different parts work well separately. This implies evaluating the rankings from both parts before making the fusion.

Ranklist	**mP@100**	**mP@50**
Faces	90.36	91.73
Faces + top20	94.53	95.80
Locations	94.10	98.53

Table 2: $mP(100)$ and $mP(50)$ for different ranking lists.

Table 2 contains P@100 and P@50 for the different systems independently. We recall that both systems for location and for faces are highly precise considering the top-100 annotations. Furthermore, we can validate that performing query expansion using the top-20 faces will lead to better rankings for faces and better final ranks.

Normalization type	**Person mP@50**	**Location mP@50**	**Final mP@50**
None	73.00	45.86	**22.26**
Max-min	68.53	42.13	13.86
Z-score	95.60	9.46	9.26
Extreme value [14]	15.80	97.06	15.46

Table 3: Different fusion strategies

Table 3 presents different fusion strategies evaluated using the interface and computing P@50. Note that best performance is obtained when no normalization is applied, so final rank is computed just by taking min value between faces and places scores per shot. This is the strategy applied for our final submissions.

2.5 Summary and Conclusions

We proposed a framework based on CNN pre-trained models to represent face and locations and a tool to annotate and evaluate different approaches by computing Precision on top-K. Performance independently generates highly precise results on top-50 and top-100, which allows us to perform query expansion by aggregating descriptors of the top-N locations or places to generate better rankings that we will combine later with late fusion. Experiments quantitatively show the effect of applying this query expansion technique on the top-20 detected faces (mAP increases from 0.031 to 0.059).

Unfortunately, after submission, we found that there was a mistake in generating the final ranked list for locations: Rankings were cropped to the top relevant 1,000 keyframes and filtered to obtain unique shots, which reduced the candidates to only 400 or less for each location. The short retrieved list explains why other normalisation types (in Table 3) that model score distributions perform worse than applying no normalisation since parameters to model localisation scores have significantly fewer samples than in the ranked list for faces. Furthermore, this is responsible for the nearly random mAP of our runs. Since candidate locations are limited to

max 400, the number of shots belonging to the intersection generated by the face and place ranks will be less then 400, and then the rest of the 1,000 submitted results will be ranked by similarity face scores that may or not contain the target place.

The simple strategy of re-sorting results using annotated data used to consistently improve results in our two last submissions, do not improve results in this submission since only a maximum of 400 results are potentially relevant among the 1,000 shots annotated by the user. Even though negative results were removed from the lists, the new ranks were filled with shots with high face score which correspond to random filling in terms of place scores and translates into a deterioration in performance (results obtained with run 3 and run 4).

3 Video to Text Description (VTT)

The VTT task was a showcase/pilot task to explore issues around how well we can automatically describe a video in natural language as we aim to move from image to video captioning. Given a set of 2,000 Vine videos, each with 2 manual captions (A and B), participants were asked to return for each video a ranked list of the text descriptions from each set, A and B. A second sub-task was to automatically generate a text description for each of the 2,000 Vine videos. The manual captions were created with a briefing for the annotators to concentrate on who is in the video, what are the objects in the video, where it is taking place and when. These captions, A and B, provided the ground truth for evaluations.

3.1 Pre-Processing Vine Videos for the VTT Tasks

Object Concepts: We used the VGG-16 deep convolutional neural network [16] to map keyframes in the videos to 1,000 object concept probabilities. We used 10 equally spaced keyframes per Vine video. The model was pre-trained on the ImageNet ILSVRC training data [13], which consists of approx 1.3 million training images in 1,000 non-overlapping categories. Keyframes were re-sized (via warping) to 224×224 before input to the network. We did not apply any data augmentation like overcropping or flipping. The output of this phase is $10 \times 1,000$ object probabilities per Vine video.

Behaviour Concepts We applied crowd behaviour recognition to categorise the motion characteristics of a given Vine sequence. Keyframes are extracted and probability scores calculated for 94 crowd behaviour concepts such as fight, run, mob, parade and protest. The mean concept score vector is then taken across the keyframes for a given Vine. These 94 concepts are taken from the WWW (Who What Where) crowd dataset [15] which contains 10,000 video sequences fully annotated for all concepts. This dataset is used to train a multi-column CNN which produces behaviour concept probability scores using both motion and appearance information. The appearance column is fed raw pixel data while the motion column is fed a dense optical flow field before the column outputs are fused and 94 concept probability scores are produced. The Alexnet architecture [6] is used for both columns. The model achieves an ROC curve AUC of 0.8656 on the WWW test set and produces a vector of concept scores than can be directly used for the VTT tasks.

Locations Locations were represented by extracting the probability scores from the softmax layer of the VGG16 network [16] pre-trained on the Places2 Dataset [21]. This dataset contains over 1.8M images from 365 different scene categories (e.g. *airport_terminal*, *cafeteria*, *hospital_room*), which makes prediction of this network very suitable for this task. The typical pre-processing for extraction of the CNN descriptor was applied to the images, resizing to 224×224 resolution.

3.2 The Caption Ranking Sub-Task

3.2.1 Methodology

We undertake a traditional information retrieval-based approach for ranking captions, treating them as retrievable documents and each video as a query. The concept description of each frame of a video comprises a fixed number of three weighted vector descriptors:

- Place descriptor of 365 dimensions, each dimension corresponding to a type of place, e.g. 'airplane', 'mansion' etc.
- Object descriptor of 1,000 dimensions, each dimension pertaining to a particular type of animate or inanimate object, e.g. 'tiger', 'gold-fish' etc.
- Action descriptor of 95 dimensions, each dimension pertaining to a type of action, e.g. 'picnic', 'marathon' etc.

The text query constructed from the descriptors comprises a list of weighted query terms, corresponding to each component of the descriptor. To obtain a single query formulation for an entire video, we simply take the average of the concept descriptor vectors over each individual image frame.

For each descriptor type, we take the text from the top-5 components. In total, the query used to rank the captions is a weighted query comprised of 15 terms (5 corresponding to place descriptor and so on). The more accurate the weights of the descriptors, the better the query representation, e.g. if for a video of an airplane take off, the weight associated with the place descriptor corresponding to the type 'airplane' is high in comparison to others, this query term dominates and the retrieval model will give higher rank to captions containing this term. The weight of each descriptor type (i.e. place, object and action) is controlled by linear combination parameters α_p (weight for place), α_o (for object), and α_a (action descriptor), which sum to 1. After some initial experiments on the training dataset of queries, we set: $\alpha_p = 0.3$, $\alpha_o = 0.4$ and $\alpha_a = 0.3$.

The retrieval model used for the caption ranking task is BM25. We set b (the length normalization parameter to 0.75 as reported to be optimal on the TREC collections [12]). We did experiments on the training data in order to find an optimum value for the BM25 parameters K (term frequency importance) of 0.8.

3.2.2 Runs Submitted

We submitted 4 runs, a BM25 baseline, a fusion run, a word2vec-based run, and a fusion between our baseline and word2vec with the performance shown in Figure 4.

The word2vec-based caption re-ranking approach relies on data from two different sources: (i)) the objects, actions and locations identified using VGG-16 (cf. Section 3.1) and (ii) automatically generated captions generated by the NeuralTalk2 system (cf. Section 3.3.1). Word similarity from word2vec [8], a Distributional Semantic Model (DSM) accessed through our Distributional Infrastructure (DInfra) [2], is used to re-rank the provided captions using these sources. The process consists of the following steps:

Generating object-action-location triples based on VGG-16 data For each video, we use the ten objects (cp. Section 3.1), actions (cp. Section 3.1) and locations (cp. Section 3.1) with the highest confidence scores. Based on these, we generate all possible object-action-location triples, resulting in a total of 1000 triples. Each of these triples then receives a score which is calculated according to Equation (2) where *rel()* is the relatedness score derived from the word2vec DSM and *conf()* denotes the confidence score assigned by the visual recogniser (averaged across keyframes). The intuition behind this scoring step is that objects, actions and locations with similar semantic contexts (as captured by the DSM relatedness measure) are more likely to co-occur in the video. We

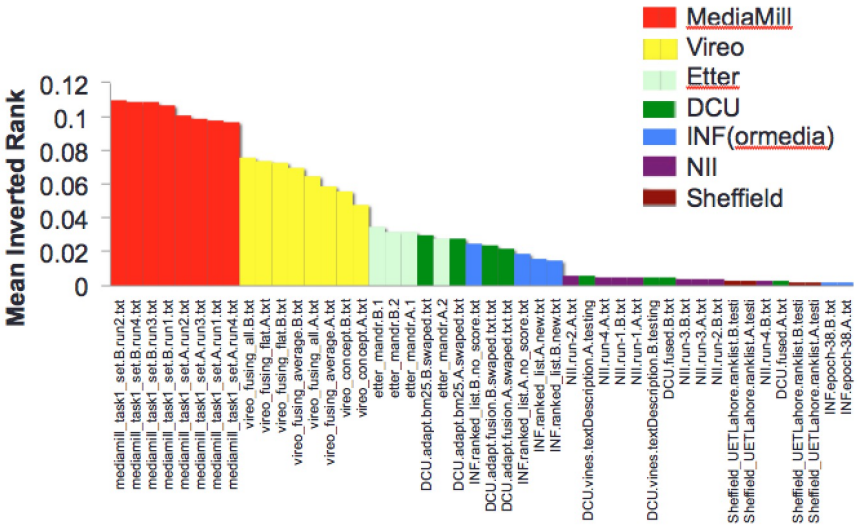

Figure 4: Scores for Caption Ranking

additionally give preference to concepts which have been identified with high confidence by VGG-16.

$$\begin{aligned} triple\_score(o,a,l) = {} & rel(a,o) * conf(a) * conf(o) + \\ & rel(o,l) * conf(o) * conf(l) + \\ & rel(a,l) * \end{aligned} \tag{2}$$

Extracting object, action and location words from automatically generated captions. In this step, we use automatically generated captions as an additional source of visual information. We run a custom text analysis pipeline using the GATE toolkit [4] to extract object, action and location words from the automatically generated captions (cf. Section 3.3.1). The pipeline applies tokenisation, POS-tagging as well as NP- and VP-chunking on each caption. A custom JAPE[4] grammar [5] is then used to to manipulate the annotations produced from this linguistic preprocessing to extract the relevant words as follows:

- locations: location expressions from a predefined list (e.g. "indoors", "outdoors") as well as nouns contained in prepositional phrases (e.g. "at the *bar*")
- actions: verbs and nouns contained in verb phrase chunks and their nominal complements (e.g. "*riding* a *skateboard*")
- objects: nouns contained in all other types of nominal chunks

Interestingly, we noted performance gains when using social media-specific tokenisation and POS tagging models from the TwitIE[5] pipeline [3], presumably due to the short and informal nature of the captions. With this structured extraction we aim to achieve better precision compared to using a simple bag-of-words approach on the captions.

[4] Java Annotations Pattern Engine

[5] Twitter Information Extraction (TwitIE) is the adaptation of the default low-level linguistic analysis resources for rule-based named entity classification in GATE to the microblogging domain.

Re-ranking candidate captions using weighted triples and words from automatically generated captions. In order to rank the candidate captions, the GATE pipeline is also applied to them to identify object, action, and location words. These words are then input to our ranking measuring the relatedness (using word2vec again) between them and a) the triples based on VGG-16 data, and b) the GATE extracted locations, actions, and objects from the automatically generated captions. In step a), a *caption_score* for each candidate caption is obtained by using the N top-ranked triples and summing the relatedness scores between the triple object & candidate caption object, triple action & candidate caption action, and triple location & candidate caption location across all of them, as in Equation (3). In case the candidate caption contains multiple objects/actions/locations words, the relatedness is averaged. The N top-ranked triples are acting as *inputone*, the candidate captions as *inputtwo*.

$$
\begin{aligned}
&\textit{input1} \;=\; [\text{io\_obj,io\_act,io\_loc}] \\
&\textit{input2} \;=\; [\text{it\_obj,it\_act,it\_loc}] \\
&\textit{caption\_score}(\textit{input}1,\, \textit{input}2) \;= \\
&\frac{1}{\textit{io\_obj.length} * \textit{it\_obj.length}} \times \sum_{ioo=1}^{\textit{io\_obj.length}} \sum_{ito=1}^{\textit{it\_obj.length}} \textit{rel}(\textit{io\_obj}[ioo], \textit{it\_obj}[ito]) \\
&+ \frac{1}{\textit{io\_act.length} * \textit{it\_act.length}} \times \sum_{ioa=1}^{\textit{io\_act.length}} \sum_{ita=1}^{\textit{it\_act.length}} \textit{rel}(\textit{io\_act}[ioa], \textit{it\_act}[ita]) \\
&+ \frac{1}{\textit{io\_loc.length} * \textit{it\_loc.length}} \times \sum_{iol=1}^{\textit{io\_loc.length}} \sum_{itl=1}^{\textit{it\_loc.length}} \textit{rel}(\textit{io\_loc}[iol], \textit{it\_loc}[itl])
\end{aligned}
\tag{3}
$$

Step b) carries out the same relatedness scoring between the object/action/location words from the automatically generated captions and the candidate captions. In addition, an averaging as in step a) is applied (Eqn 3). The automatically generated captions are acting as *inputone*, the candidate captions as *inputtwo*.

The final ranking (Eqn. 4) uses a weighted combination of the output of steps a) and b) and thus has three different parameters:

1. triple weight (α): the weight assigned to the triple score from step a)
2. top triples (N): the number of triples used for calculating the triple score in step a)
3. caption weight (β): the weight assigned to the caption score from step b)

$$
\begin{aligned}
\textit{weighted\_score}(\alpha, \beta, N) \;=\; &\alpha * \textit{caption\_score}(\textit{triples}[0:N], \textit{candidate\_caption\_words}) \\
&+ \beta * \textit{caption\_score}(\textit{auto\_generated\_caption\_words}, \textit{candidate\_caption\_words})
\end{aligned}
\tag{4}
$$

Based on experiments on a development set of 200 images and captions, we found the following parameters to be best: $\alpha = 0.8$, $N = 10$, $\beta = 0.1$.

3.3 The Caption Generation Sub-Task

3.3.1 Runs Submitted

Inspired by recent work in multimodal natural language processing such as the Multimodal Machine Translation (MMT)[6] task [18], caption generation models become a strong technique to capture and

[6] http://statmt.org/wmt16/multimodal-task.html

determine objects in the images and express their relationships in natural language. For this reason, we used an attention-based model for automatic caption generation of images extracted from the VTT videos. Despite the difficult nature of this task, recent work shows a significant improvement in the quality of caption generation by using a combination of CNNs to obtain vectorial representations into natural language sentences [20] aided by advances in deep neural networks [6].

An open sourced image-to-caption CNN-RNN toolkit – NeuralTalk2 – was used in generating captions from the extracted images.[7] NeuralTalk2 is written in Torch and it is batched and runs on a GPU. It also supports CNN finetuning, which helps with performance. In our task, we directly use the pre-trained model that was trained on the MSCOCO[8] data set. NeuralTalk2 takes an image and predicts its sentence description with a Recurrent Neural Network. Since we segmented the video into several static images, we generate one caption for each image of the video as one of the candidates for the video caption.

3.3.2 Examples

Figure 5 shows two examples of attending to the correct/wrong object and the description of the video using NeuralTalk2 on the extracted images.

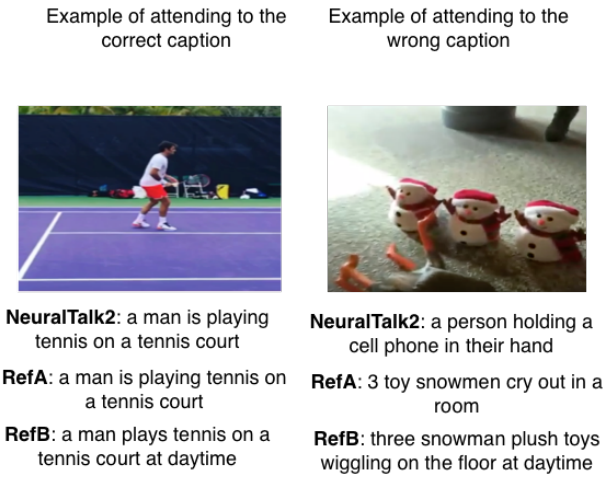

Figure 5: Examples of attending to the correct/incorrect object and description

The performance of our single submitted run is shown in Figure 6 showing it low ranked for BLEU and mid-ranked for METEOR. There are issues with the validity of the evaluation metrics used for the caption generation sub-task and documented in [1] meaning we will not analyse these results in too much detail.

4 Conclusions and Future Work

The aim of our participation in the INS task was to extend our previous work in this area while the aim of our participation in VTT was purely exploratory. Motivated by our performance

[7] https://github.com/karpathy/neuraltalk2
[8] `http://mscoco.org/`

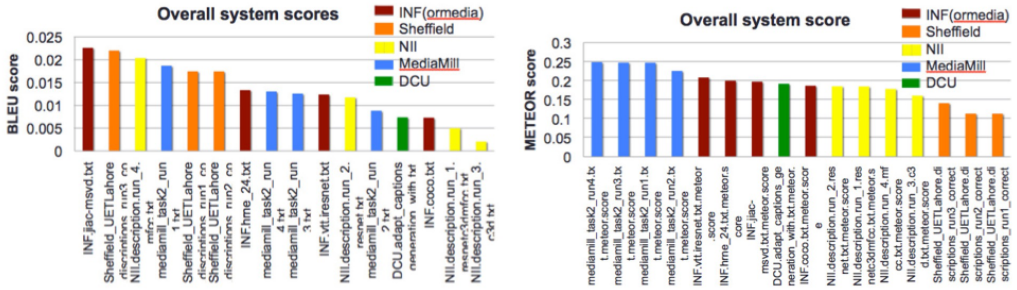

Figure 6: Scores for the Caption Generation sub-task

in caption ranking and caption generation we will refine our methods used in both tasks by broadening the number of underlying concepts.

Acknowledgement: The work reported here is based on research conducted with the support of Science Foundation Ireland under grant numbers SFI/12/RC/2289 (Insight Centre) and SFI/13/RC/2106 (ADAPT Centre).

References

[1] George Awad, Jonathan Fiscus, Martial Michel, David Joy, Wessel Kraaij, Alan F. Smeaton, Georges Quénot, Maria Eskevich, Robin Aly, and Roeland Ordelman. TRECVID 2016: Evaluating Video Search, Video Event Detection, Localization, and Hyperlinking. In *Proceedings of TRECVID 2016*. NIST, USA, 2016.

[2] Siamak Barzegar, Juliano Efson Sales, Andre Freitas, Siegfried Handschuh, and Brian Davis. Dinfra: A one stop shop for computing multilingual semantic relatedness. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1027–1028. ACM, 2015.

[3] Kalina Bontcheva, Leon Derczynski, Adam Funk, Mark A Greenwood, Diana Maynard, and Niraj Aswani. Twitie: An open-source information extraction pipeline for microblog text. In *RANLP*, pages 83–90, 2013.

[4] Hamish Cunningham, Diana Maynard, and Kalina Bontcheva. *Text processing with GATE*. Gateway Press, CA, 2011.

[5] Hamish Cunningham, Diana Maynard, and Valentin Tablan. JAPE: a Java annotation patterns engine, 1999.

[6] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[7] Kevin McGuinness, Eva Mohedano, ZhenXing Zhang, Feiyan Hu, Rami Albatal, Cathal Gurrin, Noel E. O'Connor, Alan F. Smeaton, Amaia Salvador, Xavier Giró i Nieto, and Carles Ventura. Insight Centre for Data Analytics (DCU) at TRECVid 2014: instance search and semantic indexing tasks. In *Proceedings of TRECVID 2016*. NIST, USA, 2014.

[8] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[9] Eva Mohedano, Amaia Salvador, Kevin McGuinness, Ferran Marqués, Noel E. O'Connor, and Xavier Giró i Nieto. Bags of local convolutional features for scalable instance search. *CoRR*, abs/1604.04653, 2016.

[10] Omkar M. Parkhi, Andrea Vedaldi, and Andrew Zisserman. Deep face recognition. In *British Machine Vision Conference*, 2015.

[11] Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olicier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[12] S. E. Robertson, S. Walker, and M. Beaulieu. Okapi at TREC-7: Automatic ad hoc, filtering, VLC and interactive track. In *In Proceedings of TREC 7*. National Institute of Standards and Technology (NIST), January 1999.

[13] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[14] Walter Scheirer, Anderson Rocha, Ross Micheals, and Terrance Boult. Robust fusion: Extreme value theory for recognition score normalization. In *European Conference on Computer Vision*, pages 481–495. Springer, 2010.

[15] Jing Shao, Kai Kang, Chen Change Loy, and Xiaogang Wang. Deeply learned attributes for crowded scene understanding. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4657–4666. IEEE, 2015.

[16] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[17] Alan F. Smeaton, Paul Over, and Wessel Kraaij. Evaluation Campaigns and TRECVid. In *MIR '06: Proceedings of the 8th ACM International Workshop on Multimedia Information Retrieval*, pages 321–330, New York, NY, USA, 2006. ACM Press.

[18] Lucia Specia, Stella Frank, Khalil Sima'an, and Desmond Elliott. A Shared Task on Multimodal Machine Translation and Crosslingual Image Description. In *Proceedings of the First Conference on Machine Translation, Shared Task Papers*, volume 2, page 543–553, 2016.

[19] Andrea Vedaldi and Brian Fulkerson. VLFeat: An open and portable library of computer vision algorithms. `http://www.vlfeat.org`, 2008.

[20] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of ICML 2015*, pages 2048–2057, 2015.

[21] B. Zhou, A. Khosla, A. Lapedriza, Xiao J., A. Torralba, and A. Oliva. Places: An image database for deep scene understanding. *CoRR*, abs/1610.02055, 2016.

[22] Bolei Zhou., Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using places database. In *Advances in Neural Information Processing Systems 27*, pages 487–495. Curran Associates, Inc., 2014.