Using Machine Learning Techniques, Textual and Visual Processing in Scalable Concept Image Annotation Challenge

Alexandru-Gabriel Cristea, Mădălin-Marian Savoaia, Monica-Andreea Martac, Ionela Cristina Pătraș, Alexandru-Ovidiu Scutaru, Constantin-Emilian Covrig, Adrian Iftene

UAIC: Faculty of Computer Science, "Alexandru Ioan Cuza" University, Romania
{alexandru.cristea, madalin.savoaia, monica.martac, ionela.patras, alexandru.scutaru, constantin.covrig, adiftene}@info.uaic.ro

Abstract. This paper describes UAIC[1]'s system built for participating in the Scalable Concept Image Annotation challenge 2016. We submitted runs for Subtask 1 (Image annotation and localisation), for Subtask 2 (Natural language caption generation) and for Subtask 3 (Content Selection). For the first subtask we used an ontology created last year with relations between concepts and their synonyms, hyponyms and hypernyms and also with relations between concepts and related words, but additional we used another resources and machine learning techniques. For the second subtask, we created a resource that contains triplets (*concept*$_1$, *verb*, *concept*$_2$), where *concepts* are from the list of concepts provided by the organizers and *verb* is a relation between concepts. With this resource we build sentences in which *concept*$_1$ is subject, *verb* is predicate and *concept*$_2$ is complement. For Subtask 3, we transform the input file in a form used by Subtask 2 and then we used the component developed by our team for Subtask 2.

Keywords: Machine learning, Text processing, Visual Processing, Text Generation.

1 Introduction

In 2016, UAIC group participated again in CLEF labs in few ImageCLEF tasks [1] and in this way we continued our previous participation from 2013 when we participated in Plant Identification task [2] and from 2015 when we participated in Scalable Annotation task [3]. Like in the 2015 campaign, the Scalable Concept Image Annotation challenge (from Image CLEF 2016 - Image Annotation[2]) task in 2016 aims to develop systems that receive as input an image and produce as output a prediction of which concepts are present in that image, selected from a predefined list of concepts. Similar to last year, the participants must describe images, localize the different concepts in the images and generate a description of the scene. This year the task was composed by two subtasks using a data source with 510,123 web page items

[1] University "Alexandru Ioan Cuza" of Iasi, Romania
[2] ImageCLEF 2016 – Image Annotation: http://www.imageclef.org/2016/annotation

(Subtask 1 and Subtask 2) and a data source with 10,003 entries for Subtask 3. For each item we have a corresponding web page, an image and the keywords extracted from the web page. The participants must annotate and localize concepts and/or generate sentence descriptions for all 510,123 items. More details about challenge from 2016 are in [4] and details about challenge from 2015 are in [5].

In 2014, three teams [6, 7 and 8] based their system on Convolutional Neural Networks (CNN) pre-trained using ImageNet [9]. Also, most of the teams proposed approaches based on classifiers that need to be learned [6] or based on classification with constructed ontologies [10]. In 2015, all top 4 groups used CNNs in their pipeline for the feature description [5].

The rest of the paper is structured as follows: Section 2 details the general architecture of our system, Section 3 presents the results and an error analysis, while the last Section discusses the conclusions.

2 System components

In 2016, UAIC submitted runs for image annotation and localisation (Subtask 1), for generation of a textual description of an image (Subtask 2) and for generation of a textual description avoiding processing of images (Subtask 3). For that, we built a system, consisting in modules specialized for *text processing* and *visual processing*.

2.1 Subtask 1 - Textual processing

2.1.1 Google Translate

This module is used for translating non-English words from the initial file into English for the purpose of textual concept identification. The motivation for this component is related to our built resources, which are presented in the next sections: these resources are only for English. It is using the Google Translation API v2 service[3], a file with English words and a cache file. The file with English words contains 363,802 words and we use an AVL Tree for storing them in memory to have O(log n) access time thus not wasting time. The cache file contains pairs of words in a foreign language and their English translation, for loading it in memory we use a HashMap which has a theoretical access time of O(1).

For translating a word from the initial file, we consider the following cases:

- *Case 1*: If the current word is found in the AVL Tree with English words then the program uses the word as it is and seeks to the next one from the initial file. If the word is not found then it will search for it in the cache file (Case 2).
- *Case 2*: If the current word is in the HashMap cache with translated words then the program uses the translated form and seeks to the next one from the initial file. If not it will call the Google Translate API (Case 3).

[3] Google Translation service: https://cloud.google.com/translate/v2/libraries

- *Case 3*: The program will request a translation from the Google Translate service with auto-detect as source language and English as target language and then it will use the returned result and it will also add it to the HashMap cache and file cache.

Example for Case 3 (the detected language is Spanish):

```
  Other language: ... plataformas 116 plástico 116 preventivos
116 programados 116 pública 116 tierra 116 tractores 116 técnica
116 vías 116 españa 115 madrid 113 ...

  English: ... platforms 116 plastic 116 preventive 116
scheduled 116 public 116 Earth 116 tractors 116 technique 116
way 116 Spain 115 madrid 113 ...
```

At the end of the translation process the cache file contained 935,827 pairs (*initial_word, translated_word*) and the entire process took around 19 hours to finish from an empty cache file.

2.1.2 Stop-words Elimination

This component receives a file with 510,123 lines and tries to remove every stop-word from every line with its associated number which represents its frequency. We consider additional elimination of classical stop-words (*the, from, it, he, be, is, has*, …) and the elimination of all words with one or two characters. We consider for stop-words a file with 667 entries.

For example, for input line:

```
000bRjJGbnndqJxV 100 timepiece 7988 the 4823 time 3252 or 3100
of 2664 that 2169 thesaurus 1595 for 1578 device 1569 timepieces
1513 its 1397 measuring 1286 instrument 1285 clock 1268 wheel
1232 from 1230 noun 1170 balance 1162 and 1152...
```

after using this component we obtained:

```
000bRjJGbnndqJxV 75 timepiece 7988 time 3252 thesauru 1595
device 1569 timepiece 1513 measuring 1286 instrument 1285 clock
1268 wheel 1232 noun 1170 balance 1162 legend 1105 dictionary
1095 horologe 1068 timekeeper 1063 hour 1010 keeping 978...
```

In the end, from a total of 45,771,971 words in the input file, 17,613,121 stop words were eliminated. From 2 files summing up 480.7 Mb we obtained one file with 339.6 Mb.

2.1.3 Concept Identification

Our ontology
At this step, we start to build an ontology with relations between the initial 250 concepts and related words to them (this file is based on similar file build by us in 2015 [3]). For that, we used the WordNet[4] [11] and we extracted in average around three synonyms and an average of five words that are somehow related to the concepts (automatically extracted from WordNet (hyponyms or hypernyms) and manually verified by human annotators or manually added by human annotators).

For example, for the concept *airplane* we have the following information:
Synonyms and lexical family: bombardier, landing, helicopter, jet, flight, aircraft, cabin, flying, airliner, pilot, gunner, blimp, airport, wing, fighter, terminal, hangar, cockpit, flotilla, maneuverable, passenger, alien, fly, overhead, airborne, carrier, albatross, tank, hijacking, aerospace, refuel, immigrate, aviator, fledged, airman, levitate, mobile, moth, gull, ballooning, hover, stewardess, spacecraft, butterfly, sortie, pterosaur, volant, submarine, footloose.

DuckDuckGo Ontology
Because, our ontology is still limited, in terms of concept, synonyms and relations between them (the ontology was built semi-manually, and it depends by the annotators experience and quality of work), we decided to build automatically another ontology. At this step, we start to build an ontology with related words to the initial 250 concepts. For that we used www.DuckDuckGo.com.

For example, for the concept *apple* we have the following information:
Synonyms and lexical family: cider, russet, crab, pear, stayman, fruit, quince, banana, lemon, mango, melon, orange, peach, berry, cherry, grape, plum, strudel, blackberry, currant, pomegranate, pumpkin, raspberry, strawberry, apples, almond, blueberry, carrot, cocoanut, coconut, cranberry, glacier, lemons, malus pumila, oranges, orchard apple tree, peaches, pears, potato, potatoes, turnips, alar, core, tree, eve, mac, winesap, eris, mom, pie

With this files we execute on the processed file with 510.123 lines (after steps 2.1.1 and 2.1.2) a module which identifies related concepts for every line. For that, for each word from a line, we try to find a way to connect it to concepts. That implies searching for it in the list of concepts (case 1) or in our ontology (case 2) or in DuckDuckGo ontology (case 3). If a match is found, the word is replaced with its related concept and placed in the output file along with its initial number (case 1 and case 2) or with a lower value (case 3). All words that could not be associated with any concept have been eliminated along with their number. At the next step, we sum the frequencies for the same concept for current line. After that we put in output, for

[4] WordNet: http://wordnetweb.princeton.edu/perl/webwn

current line, the initial ID and 251 bits corresponding to the list of concepts, where every bit 1 means that the corresponding concept is identified.

For example, for the following input line:

```
000bRjJGbnndqJxV 75 timepiece 7988 time 3252 thesauru 1595
device 1569 timepiece 1513 measuring 1286 instrument 1285 clock
1268 wheel 1232 noun 1170 balance 1162 legend 1105 ...
```

The output looks like:

```
000bRjJGbnndqJxV 0 0 0 0 1 0 1 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0
1 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 ...
```

Regarding the infrastructure used by our group, firstly, all programs were run on 2-core i5 Intel processors and it's took around 500 minutes. Secondly, all programs were run distributed on 5 different computers, all of them with a similar configurations, and the execution time was in average under 200 minutes.

2.2 Subtask 1 - Visual Processing

2.2.1 Face recognition

On every computer used by our team, we downloaded the archived images indexed and hashed based on the source site. For the Face Recognition part we ran the old JJIL algorithm written in Java. The results of this algorithm were either merged with the results from the textual / translation processing module or used individually with the Machine Learning Object Recognition which we are going to talk about later.

Fig. 1. An example of image that contains faces

For example for image from Fig. 1, the output after we use the JJIL API is:

```
n05600637 0.7:225x90+300+300, 0.7:855x420+300+300,
0.7:960x120+240+240, 0.7:960x120+200+200
```

We checked every image from the collection of 510,123 images downloaded from the input set. For every image, a concept ID and a position / size were generated, if the API found any faces in that image. Each couple images were structured based on the first 2 characters from their file's name.

For example, if the folder was "a5" then it contained images with file names that began with "a5 …". Our main "work" output was the textual processing / translation output. In these output files, each line had the unique Image ID followed by 251 characters of 1 and 0 (concepts). We parsed these files and for each image we easily obtained its location in the local file system through concatenation of the first 2 characters of the ID, the ID itself and the .jpg extension.

Example: `J41e3GF7p7rGdsj4 => J4 (folder) => .jpg (extension) =>`
`Path: source_folder/J4/J41e3GF7p7rGdsj4.jpg`

After obtaining the path, we wrote the subtask ID (1), the ID of the Image and the result based on the Textual Processing / Translation combined (or not) with the Face Recognition API and the Machine Learning Object Recognition.

Final output line example:

```
1 J41e3GF7p7rGdsj4 n02709367 0.7:0x0+640+427 n05563770
0.7:0x0+640+427 n02774152 0.7:0x0+640+427 ...
```

Downloading the 510,123 images took 3-4 hours for each computer. Then, the face recognition algorithm was executed on 13 computers for sets of 30,000 - 40,000 images to reduce the total duration. The face recognition API needs approximately 1 - 2 seconds for each image to identificate faces and to write the result in the output file.

2.2.2 Object recognition with Machine Learning

Object Recognition with Machine Learning was achieved through the API TensorFlow[5]. It is an API written in Python for Linux based systems that has 1,000 inner concepts created through Neural Network learning. We mapped semi-automatically these 1,000 concepts by finding a logical "connection" between them and the ImageClef's 251 concepts. The mapped file (JSON key:value) contained the TensorFlow's concepts (string) and the ImageClef's concepts ID represented by the "line position - 1" . (line position = the position of the ID in the official concepts list).

Example: `"fireboat":30, "gondola":30, "speedboat":30,`

The "30" represents n02858304 `boat.n.01`. After the mapping part, we used a Virtual Machine with a Linux Based OS to run the API that had sets of 30,000 -

[5] Tensor Flow Script: https://www.tensorflow.org/versions/r0.8/tutorials/deep_cnn/index.html

50,000 images and the mapped JSON as input. The algorithm analysed the Images and the JSON and generated an intermediate output file which contained on each line the Image ID followed by 251 characters of 1 or 0 separated by space. (if the API recognised any of its concepts in the Image, the character was 1; else, 0).

The Python script could use from 1 to 10 threads for faster execution and it took about 50 hours for a set of 50,000 images. After obtaining and mixing/ordering the parts of the intermediate results, we combined the intermediate results (ours and the textual processing one) through logical OR (1 OR 0 = 1, 0 OR 0 = 0 …), using a fast C program written by us, and afterwards we generated the final output files by using the Java algorithm.

2.3 Subtask 2 – Text Generation

For subtask 2 we are given an input file containing 510,123 lines, each of them respecting the following pattern: The line begins with the image's code, followed by a space character and after that by a list of 251 values of 0 or 1, separated by space. Each value of 0/1 from the list corresponds to a concept from the list of concepts given by the organizers: a value of 1 means that the concept belongs to appears in that image, while a value of 0 means that the concept is not in the photo.

We grouped the 251 concepts into 43 categories. For these categories, we built manually, with human annotators, a matrix that gives us the most suitable verb for linking every possible pair of 2 categories (if a suitable verb exists for 2 given categories). With all these resources we built sentences with form: ($concept_1$, *verb*, $concept_2$), in which $concept_1$ is subject, *verb* is predicate and $concept_2$ is complement.

For example, in our matrix are the following types of triplets:

- Body_part – wearing – accessories;
- Animal – drinking – drink;
- insect – in on – land vehicle; ·
- animal – near – man made object;
- animal – playing – sport_item_or_toy.

First of all, we read the entire input file and we map each line into an object containing the image's code and an array with the concepts which are present in that image. Then for every line we consider the following cases:

Case 1 - the matrix contains verbs that link the concepts in the image. In this case, a phrase that describes an image will contain maximum 3 sentences in form ($concept_1$, *verb*, $concept_2$).

Case 2 - the matrix doesn't contain verbs that link the concepts in the image. We will build sentences with the following format:
"*The image contains"/ "has"/ "includes"* + *concept*", so that a final output phrase for an image looks like: "*The image contains an X, and has a Y and includes a Z.*", where X, Y and Z are concepts from that image.
If there are duplicates among the concepts that will appear in the phrase describing the image, we will compress the phrase into:

- *The image contains 3 X's.* – if X = Y = Z

- *The image contains 2X's and the image has a Z.* – if X = Y ≠ Z.

And so on.

Case 3 - the image has not concepts. We return "Empty sentence".

Finally, the output for this subtask will look like:

```
2 000qUQAfomr0QAm4 The ribbon is on a letter, the letter is near
a bag and the house is near a garden.
2 dZ29Q2T0HBwvNQIi The image contains a telephone, has a cap and
includes a beach.
2 00hMe3sGLZSXzJKH The image contains 1 temple and 1 fan.
```

where 2 is the prefix identifying the current subtask, followed by image's code and the phrase generated by the algorithm. In this way, we have improved the rate of success in creating sentences – only the images with no concepts will cause an "Empty sentence" description.

The program used was single threaded and took about 11 hours to complete all 510,123 input lines on a dual core processor.

2.4 Subtask 3 – Content Selection

The input for Subtask 3 consist of a file having 10,003 lines, with data for 450 images. After parsing the input given, we obtained (as in the precedent subtask) a list of images, each image having a list of concepts that appears in that image. From this point on, we used the sentence generation algorithm from Subtask 2 to form phrases for each of 450 images. In order to select the triplets ($concept_1$, *verb*, $concept_2$), we consider the combination between concepts with highest frequency. Then, we consider maximum 4 triplets, in cases when we have a verb in our matrix, for considered concepts. The program used was single threaded and took about 15 minutes to run on a dual core processor.

3 Results and Evaluation

For the 2016 task, our team submitted 7 runs for Subtask 1, 2 runs for Subtask 2 and 1 run for Subtask 3 [4]. The description and duration for every run is presented in bellow Table.

Table 1: Description of runs for Subtask 1

	Description	Duration
Run 1	Based on visual processing using only the Face Recognition algorithm.	1 hour
Run 2	Based on the textual processing for the Ontology 1, 2 and the Translation.	27 hours
Run 3	Obtained through the Face Recognition and Machine Learning Object Recognition APIs.	5 days

	Description	Duration
Run 4	Mixed visual processing / textual processing output. The visual part has been made with the Face Recognition API, Machine Learning Obj. Rec.	6 days
Run 5	Textual processing output using our ontology.	7 hours
Run 6	Textual processing output using our ontology and translation	27 hours
Run 7	Textual processing output using DuckDuckGo ontology	7 hours

The translation part of the project has been made possible by the Google Translation Service[6]. The Face Recognition algorithm we used is the old one revised / optimized to analyse the images faster and safer. TensorFlow Script[7] was written so it can use multiple threads. It also had a "recheck and continue" option. This was useful in case that the Virtual Machine broke / didn't respond well so the script checked the old result saved before being closed by force and continued it.

The analysation took most of our time and computing power because it ran in a Virtual Machine. The fact that it is Python based made it even slower. Afterwards, mixing the results between Tensor Flow and Textual Processing took some more time because the lines in the intermediate output files were not in the same order so we used Linux's bash commands for sets difference, sorting, cut-in etc. to fix the problem.

Finally, the final output files were obtained running the Java algorithm with or without the Face Recognition option and later we validated them with the official bash script.

Table 2: Description of runs for Subtask 2

	Description	Duration
Run 8	Based on Cases 1, 2, 3 presented in 2.3, using like input file the Run 2	11 hours
Run 9	Similar to Run 8, with input file Run 4	11 hours

In the case of Subtask 2, we submitted two runs with different files like input from Subtask 1. We consider for input files two cases: Run 2 and Run 4 from Subtask 1. For Subtask 3 we submitted Run 10, which took around 15 minutes.

3.1 Evaluation for Subtask 1

Table 3 below gives the results for the runs from Subtask 1 described above. More details are in [4].

Table 3: Results of UAIC's runs from Subtask 1

% Overlap with GT labels	R1	R3	R4
50 %	0.001526	0.001526	0.001526
0 %	0.00281	0.00281	0.00281

[6] Google Translation service: https://cloud.google.com/translate/docs
[7] Tensor Flow Script: https://www.tensorflow.org/versions/r0.8/tutorials/deep_cnn/index.html

As we can see from Table 3, the R1, R3 and R4 have the same results. We didn't add to this table the rest of runs, because the values for them are 0.

3.2 Evaluation for Subtask 2 and Subtask 3

Tables 4 and 5 from below give the results for the runs from Subtask 2noisy track and Subtask 2 clean track described above.

Table 4: Results of UAIC's runs from Subtask 2

	R8	**R9**
MEAN	0.0896	0.0934
STDDEV	0.0297	0.0249
MEDIAN	0.0870	0.0915
MIN	0.0161	0.0194
MAX	0.2230	0.2514

Table 5: Results of UAIC's runs from Subtask 3

	R10
Mean F	0.4982 +- 0.1782
Mean P	0.4597 +- 0.1553
Mean R	0.5951 +- 0.2592

We can see how R9 is better than R8. For R10, we remark positive the good value for precision, but also we remark negative the lower value for recall.

4 Conclusions

This document details the system developed by the UAIC team for the 2016 edition of ImageClef's contest on Image Annotation (Face Recognition). The entire system has modules for each of the 3 subtasks: Concepts Detection and Annotation (Subtask 1), Natural Language Generation (Subtask 2) and Bounding Boxes Concepts Identification (Subtask 3).

The Subtask 1 is the main component of the system and consists in textual processing of each website's content (which also means translating words that are not in English) and concept identification, and also visual processing. This one represents Face Recognition and Object Recognition through Machine Learning.

For Subtask 2 and for Subtask 3, we used a matrix built by us, with relations between concepts in form (*concept1, verb, concept2*), in which $concept_1$ is subject, *verb* is predicate and $concept_2$ is complement. When concepts cannot be linked using this matrix, we use a generic phrase in which the concepts are enumerated.

For further development of the system we think that finding APIs and adapting them in a more thoughtful and efficient manner to identify concepts is an important target. We should aim to have more concepts identified more precisely for each

image. As for the translation part we think that the Google Translation Service works well enough for the moment, even though it takes some time to finish “its jobs”.

Acknowledgement. Special thanks go to all colleagues from the Faculty of Computer Science, second year, group B2, who were involved in this project.

References

1. Villegas, M., Henning, M., Seco de Herrera, Alba, G., Roger, S., Stefano, B., Gilbert, A., Piras, L., Wang, J., Yan, F., Ramisa, A., Dellandrea, E., Gaizauskas, R., Mikolajczyk, K., Puigcerver, J., Toselli, A., Sánchez, J.A., Vidal, E. General Overview of ImageCLEF at the CLEF 2016 Labs. Springer International Publishing, Lecture Notes in Computer Science, ISSN 0302-9743. (2016)
2. Șerban, C., Sirițeanu, A., Gheorghiu, C., Iftene, A., Alboaie, L., Breabăn, M. Combining image retrieval, metadata processing and naive Bayes classification at Plant Identification 2013. Notebook Paper for the CLEF 2013 LABs Workshop - ImageCLEF - Plant Identification, 23-26 September, Valencia, Spain. (2013)
3. Calfa, A., Silion, D., Bursuc, A. C., Acatrinei, C. P., Lupu, R. I., Cozma, A. E., Pădurariu, C., Iftene, A. 2015. Using Textual and Visual Processing in Scalable Concept Image Annotation Challenge. In Working Notes of CLEF 2015 - Conference and Labs of the Evaluation forum - ImageCLEF2015. Vol. 1391, ISSN 1613-0073. (2015)
4. Gilbert, A., Piras, L., Wang, J., Yan, F., Ramisa, A., Dellandrea, E., Gaizauskas, R., Villegas, M., Mikolajczyk, K. Overview of the ImageCLEF 2016 Scalable Concept Image Annotation Task. CLEF2016 Working Notes, CEUR Workshop Proceedings. (2016)
5. Gilbert, A., Piras, L., Wang, J., Yan, F., Dellandrea, E., Gaizauskas, R., Villegas, M., Mikolajczyk, K.: Overview of the ImageCLEF 2015 Scalable Image Annotation, Localization and Sentence Generation task. In CLEF2015 Working Notes – CEUR Workshop Proceedings, Publisher CEUR-WS.org, ISSN: 1613-0073.Toulouse, France, September 8-11. (2015)
6. Villegas, M., Paredes, R.: Overview of the ImageCLEF 2014 Scalable Concept Image Annotation Task. In: CLEF 2014 Evaluation Labs and Workshop, Online Working Notes. (2014)
7. Kanehira, A., Hidaka, M., Mukuta, Y., Tsuchiya, Y., Mano, T., Harada, T.: MIL at ImageCLEF 2014: Scalable System for Image Annotation. In CLEF 2014 Evaluation Labs and Workshop, Online Working Notes. Sheffield, UK, September 15-18. (2014)
8. Vanegas, J.A., Arevalo, J., Otálora, S., Páez, F., Pérez-Rubiano, S.A., González, F. A.: MindLab at ImageCLEF 2014: Scalable Concept Image Annotation. In CLEF 2014 Evaluation Labs and Workshop, Online Working Notes. Sheffield, UK, September 15-18. (2014)
9. Xu, X., Shimada, A., ichiro Taniguchi, R.: MLIA at ImageCLEF 2014 Scalable Concept Image Annotation Challenge. In: CLEF 2014 Evaluation Labs and Workshop, Online Working Notes. Sheffield, UK, September 15-18. (2014)
10. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: A large scale hierarchical image database. In Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on. pp. 248–255. June. (2009), doi:10.1109/CVPR.2009.5206848
11. Fellbaum, C.: WordNet: An Electronic Lexical Database. Cambridge, MA: MIT Press. (1998)