Automatic Text Summarization for Annotating Images

Gediminas Bertasius

November 24, 2013

1 Introduction

With an explosion of image data on the web, automatic image annotation has become an important area of machine learning, computer vision and natural language processing research. The goal of automatic image annotation systems is to generate the key words or sentences that capture the most important content in the image. There are several ways how to approach this problem. The most typical way is to employ various computer vision techniques to analyze the image, which we want to annotate. Another, completely different approach is to utilize the text provided with the image and try to capture the most important ideas in the text and use those ideas to generate the annotations.

In my project, I focused on the latter technique. I utilized textual information that was accompanied with an image to infer most likely words that could be used in the caption of that image. I was operating under the assumption that captions of an image are highly correlated with the most important information in the text. Given that I was using BBC news dataset, this was a perfectly reasonable assumption to make.

In my project, I experimented with several different approaches. Initially, I implemented tf-idf text representation and used it as a baseline to evaluate my proposed methods performance. My proposed methods consisted of two discriminative models and one generative model. As my discriminative models, I utilized Sentence-Feature and Word-Feature models, which will be described in the later sections. Such a representation allowed me to transform annotation problem into a classification problem, which was much more convenient. For my generative model, I utilized Hidden Markov Model, an idea similar to the one presented in [1].

2 Related Work

As already mentioned previously, there are two common ways to approach image annotation problem: from the computer vision perspective and from the natural language processing perspective. Because annotation problem is more directly linked to the images, computer vision techniques have been more popular at this task in the past. However, as natural language processing algorithms got more sophisticated there has been an increased number of attempts to approach
Figure 1: Examples of what an annotation task looks like. Given an image or a text (or both) an annotation system has to generate a caption for the image. Examples of captions are given in the bounded red boxes in the figures above this problem using natural language processing methods. In this section I will briefly describe past work on image annotation in both fields.

Since image annotation is directly linked to the analysis of image contents, many computer vision scientists tackled this problem [4] [7] [8]. Two most popular techniques to approach this problem includes object classification and image segmentation. Image annotation can be simply viewed as an object classification problem with very large number of classes. Hence, all of the methods applied to object categorization would be also applicable to image annotation. Another way to approach image annotation problem is to segment the image into separate regions and associate a specific word with a certain region, which may seem more intuitive but also more difficult to implement in practice.

Additionally, because images on the web are also usually accompanied by large amounts of text, image annotation problem has also been explored in the field of natural language processing [3] [5]. Because image annotation problem in natural language processing is still an emerging area there have not been very well defined methods for this particular task. Currently most common methods include tf-idf, Latent Dirichlet Allocation [2], or simply using words from the title to generate captions for the image.

3 Dataset

For my project I used BBC news dataset [5]. This dataset includes 3121 training and 240 testing samples respectively. Each data instance includes an article, an image associated with an article and the caption under the image. This dataset is highly applicable to the methods in both computer vision and natural language processing, which is highly beneficial for the comparison purposes between the two.
4 Methods

4.1 tf-idf

As a baseline method, I employed tf-idf text representation with logarithmically scaled weights. tf-idf scheme is defined as follows:

$$tf(t, d) = \log(1 + f(t, d))$$

$$idf(t, D) = \log \frac{|D|}{|\{d \in D : t \in d\}|}$$

where $f(t, d)$ denotes frequency of a term $t$ in document $d$ and $D$ refers to the number of documents in the corpus. Then these two measures are combined to compute $tf$-$idf$ weight in a following way:

$$tf-idf(t, d, D) = tf(t, d) \times idf(t, D)$$

Intuitively it makes sense that words, which appear more frequently in the text are more likely to be used in the annotations. tf-idf representation would capture this idea and thus, should serve well as a baseline measure to evaluate the relative performance of my proposed methods.

4.2 Sentence-Features Model

4.2.1 Description

The basic idea behind this method is to find most salient sentences in the text and then utilize most prominent words from these sentences to generate captions for the images. My rationale for this method is that each text contains several sentences that capture the most important ideas in the text. As a result, the ideas from these sentences should be much more likely to be used as captions under the images. In this particular case, I am making an assumption that these ideas will be expressed using similar words. Even though this assumption may not necessarily hold true in all cases, this is the best we can do for the moment.

4.2.2 Features

The idea is to transform each sentence to a feature vector that would capture the relationship between the semantic content in a specific sentence and the rest of the text. It is reasonable to assume that most important sentences in the text will share similar ideas with many other sentences. To capture this relationship I used two different ideas for feature construction. First, I incorporated sentence position as one of the features because it is natural that sentences at the beginning or at the end of the text contain more important ideas. In addition, I utilized "word2vec" deep learning toolkit [6], which converts each word into its vector representation. Then to create the feature for the entire sentence, I computed cosine similarities between the words in the current sentence and the words in the rest of the text. Finally, I created a histogram of 20 bins out of these similarities and used this as my feature vector.
4.2.3 Labels

The labels are designed in a similar fashion as the features. For each word in a given sentence I compute cosine similarity between that word and each of the words in a given summary of the article. I then take the average of all the similarities and use it as my final label. At the end, instead of a binary classification this turns out to be a regression problem.

4.2.4 Classifier

To classify my feature vectors I utilized gradient boosted decision trees algorithm [9]. Gradient boosted decision trees have been shown to yield good classification results on datasets that involve textual features [10]. Therefore, it seemed appropriate to use this classifier for my particular task.

4.3 Word-Features Model

4.3.1 Description

As opposed to treating the entire sentence as my feature vector like I did in the previous method 4.2, for my other method I decided to experiment with the representation where each word represents a feature vector by itself. There are several reasons for this representation. First, creating a features vector for each word is much easier task than creating a feature vector for every sentence because each sentence varies in length. Secondly, in theory such a representation should make classification a bit more accurate. This is because our goal essentially is to predict most important words rather than sentences. Therefore in the training stage, word feature representation allows the classifier to learn the intrinsic properties of the words that are used for the captions.

4.3.2 Feature Representation

I utilized three different ideas to construct my feature vector for each word. First, as one of the entries I incorporated sentence position in the text, in which that word appears. As described in section 4.2, because sentence position in the text may signify its importance this is an important factor to consider. To incorporate semantic similarity between the word and the rest of the text I utilized the same deep learning toolkit ‘word2vec’ [6], which produced 200 entry vector for each of the words [6]. Finally, I concatenated tf-idf weight to the feature vector to explicitly model word frequency in the entire corpus.

4.3.3 Labels

For the labels in this model I simply used binary entries to model whether a particular word in some given text appears in the caption associated with that text or not.

4.3.4 Classifier

Unlike in my previous method, in this case I was dealing with a binary classification problem. Nevertheless, just like in the previous case, for my classifier, I used gradient boosted decision tree. However, in this case, for the decision
tree learning procedure, I employed loss function designed for binary classification problems rather than regression problems, which turned out to be working pretty well.

4.4 Hidden Markov Model

4.4.1 Description

To implement this model I used an idea similar to the one presented in [1]. To represent each sentence as a feature vector I used an identical representation as described in section 4.2. Then, I created a set of topics over all of the documents, which would correspond to the emissions in a regular HMM model. These topics were generated by applying k-means clustering algorithm to a set of sentence feature vectors. Each topic was represented as one of the final k-means clusters. After this learning procedure each sentence had some topic assigned to it.

4.4.2 Transitions and Emissions

To create my HMM model, I defined each hidden state as a sentence either belonging to the annotation or not. For the emissions I simply used topics which were represented by k-means clusters. According to my specification, each sentence emits a topic, which corresponds to our observation. All of the transition and emission probabilities were extracted from the training data.

4.4.3 Inference

Finally, to infer hidden states from the given observations I used Viterbi algorithm.

5 Evaluation

5.1 Evaluation Metric

To quantify the results of all the methods, I utilized evaluation metrics that are commonly used in information retrieval. These include precision, recall, and F-score. At a high level, precision signifies the probability that a retrieved word will be relevant for the annotation. Its formal definition is presented below:

\[
precision = \frac{|\{\text{retrieved words}\} \cap \{\text{relevant words}\}|}{|\{\text{retrieved words}\}|}
\]

The term recall refers to the fraction of relevant words that will be retrieved by the system and is defined as follows:

\[
recall = \frac{|\{\text{retrieved words}\} \cap \{\text{relevant words}\}|}{|\{\text{relevant words}\}|}
\]

Then combining these two metrics produces our final evaluation metric commonly referred to as F-score, which is defined as:

\[
F = 2 \cdot \frac{precision \cdot recall}{precision + recall}
\]
5.2 Results

The results below indicate that all of my proposed methods work better than tf-idf baseline. This makes sense because my proposed models utilize prior information about the problem in the training procedure whereas tf-idf is more of an unsupervised approach to detect annotations. Below is the figure illustrating combined precision, recall and F-score results for all of the methods.

![Figure 2: Precision, Recall and F-score measures for all of the methods](image)

As illustrated in Figures 2, 3, because Sentence-Feature model and HMM operate on the sentence rather than the word level, both of them tend to retrieve many more words than necessary, thus producing very high recall. Word-Feature model on the other hand produces more balanced results between precision and recall. Overall, however, F-scores are what matter the most. Individual F-scores for each method are presented in Figure 4. As already mentioned, F-scores for all of my methods are higher than the F-score for tf-idf text representation. These results clearly illustrate the benefit of utilizing prior information about the problem. In my first two proposed methods this is done by training gradient boosted trees on the training data, whereas in HMM prior information is used when extracting transition and emission parameters from the training data.

Because my proposed Sentence-Feature model has a dependency on a parameter controlling how many words will be selected from each predicted sentence, I also present the results illustrating model’s behavior as this parameter is varied.

Unsurprisingly, recall tends to increase as we increase number of words retrieved from the predicted sentences whereas the precision starts decreasing. F-score captures the optimal balance between these two evaluation metrics, and is therefore a fair metric to evaluate the performance of all methods.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Precision</th>
<th>Recall</th>
<th>F-Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>tf-idf</td>
<td>0.0275</td>
<td>0.1170</td>
<td>0.04453</td>
</tr>
<tr>
<td>Sentence-Feature model</td>
<td>0.05172</td>
<td>0.44153</td>
<td>0.0926</td>
</tr>
<tr>
<td>Word-Feature model</td>
<td>0.06305</td>
<td>0.1693545</td>
<td>0.09189</td>
</tr>
<tr>
<td>HMM</td>
<td>0.0453</td>
<td>0.50389</td>
<td>0.08328</td>
</tr>
</tbody>
</table>

Figure 3: Precision, Recall and F-score measures for all of the methods
5.3 Results Visualization

Furthermore, below I presented some of the results from BBC news test dataset illustrating my methods’ performance in practice. Figure 8 depict the performance of a Word-Feature model. From this example it is clear that the system successfully picks up correct proper nouns for the annotations. This makes sense because proper nouns that appear in a text usually have very specific connotations. Thus, they are much more likely to appear in the annotations than the regular words.

In addition, as illustrated in the Figure 8 the system also manages to detect some common words that are important in the given context. Obviously the task of detecting important ordinary words is a much more challenging, hence the lower accuracy in comparison to proper noun detection.

Furthermore, below I also presented some results that illustrate the performance of my Hidden Markov Model. As Figure 9 suggests, HMM generated annotations tend to retrieve more words than needed. Another important thing to notice is that the semantic meanings between HMM generated annotations and actual annotations are very similar. However, due to the use of different vocabulary in the actual annotations, HMM is not always able to detect the correct words. However, HMM’s property of producing annotations that are semantically very similar to the original annotations could definitely be utilized in text summarization. In addition, this property may also be beneficial to build more complex models that operate on the semantical level.
6 Conclusions and Future Work

Overall, given the limitation of the models, which I used in this project I am satisfied with their performances. As results illustrate, Word-Feature model was able to successfully detect most salient words in the original annotations. Additionally, both Sentence-Feature and HMM models also produced good results and successfully picked up semantically meaningful sentences to be used as the annotations.

There are couple of ways how to increase the accuracy of these models
though. First, instead of relying on syntactic and word-level models, it would be more beneficial to use models that operate on the semantics level. After capturing the most important ideas in the text, it would be much easier to detect the words associated with these ideas.

Furthermore, to utilize all of the available information it would be beneficial to combine methods from both natural language processing and computer vision. This would allow accurate annotation because some of the captions under the images are more heavily associated with the images whereas others are linked more directly to the specific ideas from the text. Joint computer vision and natural language processing system would be able to handle both of these cases thus, producing better performance.

References


