Generating image descriptions using dependency relational patterns

Ahmet Aker
University of Sheffield
a.aker@dcs.shef.ac.uk

Robert Gaizauskas
University of Sheffield
r.gaizauskas@dcs.shef.ac.uk

Abstract

This paper presents a novel approach to automatic captioning of geo-tagged images by summarizing multiple web-documents that contain information related to an image's location. The summarizer is biased by dependency pattern models towards sentences which contain features typically provided for different scene types such as those of churches, bridges, etc. Our results show that summaries biased by dependency pattern models lead to significantly higher ROUGE scores than both n-gram language models reported in previous work and also Wikipedia baseline summaries. Summaries generated using dependency patterns also lead to more readable summaries than those generated without dependency patterns.

1 Introduction

The number of images tagged with location information on the web is growing rapidly, facilitated by the availability of GPS (Global Position System) equipped cameras and phones, as well as by the widespread use of online social sites. The majority of these images are indexed with GPS coordinates (latitude and longitude) only and/or have minimal captions. This typically small amount of textual information associated with the image is of limited usefulness for image indexing, organization and search. Therefore methods which could automatically supplement the information available for image indexing and lead to improved image retrieval would be extremely useful.

Following the general approach proposed by Aker and Gaizauskas (2009), in this paper we describe a method for automatic image captioning or caption enhancement starting with only a scene or subject type and a set of place names pertaining to an image – for example $\langle$church, $\{$St. Paul's,London$\}\rangle$. Scene type and place names can be obtained automatically given GPS coordinates and compass information using techniques such as those described in Xin et al. (2010) – that task is not the focus of this paper.

Our method applies only to images of static features of the built or natural landscape, i.e. objects with persistent geo-coordinates, such as buildings and mountains, and not to images of objects which move about in such landscapes, e.g. people, cars, clouds, etc. However, our technique is suitable not only for image captioning but in any application context that requires summary descriptions of instances of object classes, where the instance is to be characterized in terms of the features typically mentioned in describing members of the class.

Aker and Gaizauskas (2009) have argued that humans appear to have a conceptual model of what is salient regarding a certain object type (e.g. church, bridge, etc.) and that this model informs their choice of what to say when describing an instance of this type. They also experimented with representing such conceptual models using n-gram language models derived from corpora consisting of collections of descriptions of instances of specific object types (e.g. a corpus of descriptions of churches, a corpus of bridge descriptions, and so on) and reported results showing that incorporating such n-gram language models as a feature in a feature-based extractive summarizer improves the quality of automatically generated summaries.

The main weakness of n-gram language models is that they only capture very local information about short term sequences and cannot model long distance dependencies between terms. For example one common and important feature of object descriptions is the simple specification of the object type, e.g. the information that the object *London Bridge* is a *bridge* or that the *Rhine* is a *river*. If this information is expressed as in the first line of Table 1, n-gram language models are likely to

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1250–1258,
Uppsala, Sweden, 11-16 July 2010.

Table 1: Example of sentences which express the type of an object.

London Bridge is a **bridge**...
The **Rhine** (German: Rhein; Dutch: Rijn; French: Rhin; Romansh: Rain; Italian: Reno; Latin: Rhenus West Frisian Ryn) is one of the longest and most important **rivers** in Europe...

reflect it, since one would expect the tri-gram *is a bridge* to occur with high frequency in a corpus of bridge descriptions. However, if the type predication occurs with less commonly seen local context, as is the case for the object *Rhine* in the second row of Table 1 – *most important rivers* – n-gram language models may well be unable to identify it.

Intuitively, what is important in both these cases is that there is a predication whose subject is the object instance of interest and the head of whose complement is the object type: *London Bridge ... is ... bridge* and *Rhine ... is ... river*. Sentences matching such patterns are likely to be important ones to include in a summary. This intuition suggests that rather than representing object type conceptual models via corpus-derived language models as do Aker and Gaizauskas (2009), we do so instead using *corpus-derived dependency patterns*.

We pursue this idea in this paper, our hypothesis being that information that is important for describing objects of a given type will frequently be realized linguistically via expressions with the same dependency structure. We explore this hypothesis by developing a method for deriving common dependency patterns from object type corpora (Section 2) and then incorporating these patterns into an extractive summarization system (Section 3). In Section 4 we evaluate the approach both by scoring against model summaries and via a readability assessment. Since our work aims to extend the work of Aker and Gaizauskas (2009) we reproduce their experiments with n-gram language models in the current setting so as to permit accurate comparison.

Multi-document summarizers face the problem of avoiding redundancy: often, important information which must be included in the summary is repeated several times across the document set, but must be included in the summary only once. We can use the dependency pattern approach to address this problem in a novel way. The common approach to avoiding redundancy is to use a text similarity measure to block the addition of a further sentence to the summary if it is too similar to one already included. Instead, since specific dependency patterns express specific types of information we can group the patterns into groups expressing the same type of information and then, during sentence selection, ensure that sentences matching patterns from different groups are selected in order to guarantee broad, non-redundant coverage of information relevant for inclusion in the summary. We report work experimenting with this idea too.

Table 2: Object types and the number of articles in each object type corpus. Object types which are bold are covered by the evaluation image set.

village 39970, school 15794, city 14233, organization 9393, **university** 7101, **area** 6934, **district** 6565, airport 6493, **island** 6400, **railway station** 5905, **river** 5851, company 5734, **mountain** 5290, **park** 3754, **college** 3749, **stadium** 3665, **lake** 3649, **road** 3421, country 3186, **church** 3005, way 2508, **museum** 2320, **railway** 2093, **house** 2018, arena 1829, field 1731, club 1708, shopping centre 1509, highway 1464, **bridge** 1383, **street** 1352, **theatre** 1330, bank 1310, property 1261, **hill** 1072, **castle** 1022, forest 995, court 949, hospital 937, peak 906, bay 899, **skyscraper** 843, **valley** 763, **hotel** 741, **garden** 739, **building** 722, market 712, **monument** 679, port 651, sea 645, **temple** 625, **beach** 614, **square** 605, store 547, campus 525, **palace** 516, **tower** 496, cemetery 457, **volcano** 426, **cathedral** 402, **glacier** 392, **residence** 371, dam 363, **waterfall** 355, **gallery** 349, **prison** 348, **cave** 341, **canal** 332, restaurant 329, path 312, observatory 303, **zoo** 302, coast 298, **statue** 283, **venue** 269, **parliament** 258, shrine 256, desert 248, synagogue 236, bar 229, **ski resort** 227, arch 223, landscape 220, **avenue** 202, casino 179, farm 179, seaside 173, waterway 167, tunnel 167, ruin 166, **chapel** 165, **observation wheel** 158, **basilica** 157, woodland 154, wetland 151, cinema 144, **gate** 142, **aquarium** 136, entrance 136, **opera house** 134, **spa** 125, shop 124, **abbey** 108, **boulevard** 108, **pub** 92, bookstore 76, **mosque** 56

2 Representing conceptual models

2.1 Object type corpora

We derive n-gram language and dependency pattern models using object type corpora made available to us by Aker and Gaizauskas. Aker and Gaizauskas (2009) define an object type corpus as a collection of texts about a specific static object type such as *church, bridge, etc.* Objects can be named locations such as *Eiffel Tower*. To refer to such names they use the term *toponym*. To build such object type corpora the authors categorized Wikipedia articles places by object type. The object type of each article was identified automatically by running *Is-A* patterns over the first five sentences of the article. The authors report 91% accuracy for their categorization process. The most populated of the categories identified (in total 107 containing articles about places around the world) are shown in Table 2.

2.2 N-gram language models

Aker and Gaizauskas (2009) experimented with uni-gram and bi-gram language models to capture the features commonly used when describing an object type and used these to bias the sentence selection of the summarizer towards the sentences that contain these features. As in Song and Croft (1999) they used their language models in a gener-

ative way, i.e. they calculate the probability that a sentence is generated based on a n-gram language model. They showed that summarizer biased with bi-gram language models produced better results than those biased with uni-gram models. We replicate the experiments of Aker and Gaizauskas and generate a bi-gram language model for each object type corpus. In later sections we use *LM* to refer to these models.

2.3 Dependency patterns

We use the same object type corpora to derive dependency patterns. Our patterns are derived from dependency trees which are obtained using the Stanford parser[1]. Each article in each object type corpus was pre-processed by sentence splitting and named entity tagging[2]. Then each sentence was parsed by the Stanford dependency parser to obtain relational patterns. As with the chain model introduced by Sudo et al. (2001) our relational patterns are concentrated on the verbs in the sentences and contain *n+1* words (the verb and *n* words in direct or indirect relation with the verb). The number *n* is experimentally set to two words.

For illustration consider the sentence shown in Table 3 that is taken from an article in the *bridge* corpus. The first two rows of the table show the original sentence and its form after named entity tagging. The next step in processing is to replace any occurrence of a string denoting the object type by the term "OBJECTTYPE" as shown in the third row of Table 3. The final two rows of the table show the output of the Stanford dependency parser and the relational patterns identified for this example. To obtain the relational patterns from the parser output we first identified the verbs in the output. For each such verb we extracted two further words being in direct or indirect relation to the current verb. Two words are directly related if they occur in the same relational term. The verb *built-4*, for instance, is directly related to *DATE-6* because they both are in the same relational term *prep-in(built-4, DATE-6)*. Two words are indirectly related if they occur in two different terms but are linked by a word that occurs in those two terms. The verb *was-3* is, for instance, indirectly related to *OBJECTTYPE-2* because they are both in different terms but linked with *built-4* that occurs in both terms. E.g. for the term *nsubjpass(built-4, OBJECTTYPE-2)* we use the verb *built* and extract patterns based on this. *OBJECTTYPE* is in direct relation to *built* and *The* is in indirect relation to *built* through *OBJECTTYPE*. So a pattern from these relations is *The OBJECTTYPE built*. The next pattern extracted from this term is *OBJECTTYPE was built*. This pattern is based on direct relations. The verb *built* is in direct relation to *OBJECTTYPE* and also to *was*. We continue this until we cover all direct relations with *built* resulting in two more patterns (*OBJECTTYPE built DATE* and *OBJECTTYPE built W*). It should be noted that we consider all direct and indirect relations while generating the patterns.

Table 3: Example sentence for dependency pattern.

Original sentence: The bridge was built in 1876 by W. W.
After NE tagging: The bridge was built in DATE by W. W.
Input to the parser: The OBJECTTYPE was built in DATE by W. W.
Output of the parser: *det(OBJECTTYPE-2, The-1), nsubjpass(built-4, OBJECTTYPE-2), auxpass(built-4, was-3), prep-in(built-4, DATE-6), nn(W-10, W-8), agent(built-4, W-10)*
Patterns: The OBJECTTYPE built, OBJECTTYPE was built, OBJECTTYPE built DATE, OBJECTTYPE built W, was built DATE, was built W

Following these steps we extracted relational patterns for each object type corpus along with the frequency of occurrence of the pattern in the entire corpus. The frequency values are used by the summarizer to score the sentences. In the following sections we will use the term *DpM* to refer to these dependency pattern models.

2.3.1 Pattern categorization

In addition to using dependency patterns as models for biasing sentence selection, we can also use them to control the kind of information to be included in the final summary (see Section 3.2). We may want to ensure that the summary contains a sentence describing the object type of the object, its location and some background information. For example, for the object *Eiffel Tower* we aim to say that it is a tower, located in Paris, designed by Gustave Eiffel, etc. To be able to do so, we categorize dependency patterns according to the type of information they express.

We manually analyzed human written descriptions about instances of different object types and recorded for each sentence in the descriptions the kind of information it contained about the object. We analyzed descriptions of 310 different objects where each object had up to four different human written descriptions (Section 4.1). We categorized the information contained in the descriptions into

[1] http://nlp.stanford.edu/software/lex-parser.shtml

[2] For performing shallow text analysis the OpenNLP tools (http://opennlp.sourceforge.net/) were used.

the following categories:

- **type**: sentences containing the "type" information of the object such as *XXX* is a *bridge*
- **year**: sentences containing information about when the object was built or in case of mountains, for instance, when it was first climbed
- **location**: sentences containing information about where the object is located
- **background**: sentences containing some specific information about the object
- **surrounding**: sentences containing information about what other objects are close to the main object
- **visiting**: sentences containing information about e.g. visiting times, etc.

We also manually assigned each dependency pattern in each corpus-derived model to one of the above categories, provided it occurred five or more times in the object type corpora. The patterns extracted for our example sentence shown in Table 3, for instance, are all categorized by *year* category because all of them contain information about the foundation date of an object.

3 Summarizer

We adopted the same overall approach to summarization used by Aker and Gaizauskas (2009) to generate the image descriptions. The summarizer is an extractive, query-based multi-document summarization system. It is given two inputs: a toponym associated with an image and a set of documents to be summarized which have been retrieved from the web using the toponym as a query. The summarizer creates image descriptions in a three step process. First, it applies shallow text analysis, including sentence detection, tokenization, lemmatization and POS-tagging to the given input documents. Then it extracts features from the document sentences. Finally, it combines the features using a linear weighting scheme to compute the final score for each sentence and to create the final summary. We modified the approach to feature extraction and the way the summarizer acquires the weights for feature combination. The following subsections describe how feature extraction/combination is done in more detail.

3.1 Feature Extraction

The original summarizer reported in Aker and Gaizauskas (2009) uses the following features to score the sentences:

- ***querySimilarity***: Sentence similarity to the query (toponym) (cosine similarity over the vector representation of the sentence and the query).
- ***centroidSimilarity***: Sentence similarity to the centroid. The centroid is composed of the 100 most frequently occurring non stop words in the document collection (cosine similarity over the vector representation of the sentence and the centroid).
- ***sentencePosition***: Position of the sentence within its document. The first sentence in the document gets the score 1 and the last one gets $\frac{1}{n}$ where *n* is the number of sentences in the document.
- ***starterSimilarity***: A sentence gets a binary score if it starts with the query term (e.g. *Westminster Abbey*, *The Westminster Abbey*, *The Westminster* or *The Abbey*) or with the object type, e.g. *The church*. We also allow gaps (up to four words) between *the* and the query to capture cases such as *The most magnificent Abbey*, etc.
- ***LMSim***[3]: The similarity of a sentence *S* to an n-gram language model *LM* (the probability that the sentence *S* is generated by *LM*).

In our experiments we extend this feature set by two dependency pattern related features: *DpMSim* and *DepCat*.

DpMSim is computed in a similar fashion to *LMSim* feature. We assign each sentence a dependency similarity score. To compute this score, we first parse the sentence on the fly with the Stanford parser and obtain the dependency patterns for the sentence. We then associate each dependency pattern of the sentence with the occurrence frequency of that pattern in the dependency pattern model (*DpM*). *DpMSim* is then computed as given in Equation 1. It is a sum of all occurrence frequencies of the dependency patterns detected in a sentence *S* that are also contained in the *DpM*.

$$DpMSim(S, DpM) = \sum_{p \in S} f_{DpM}(p) \quad (1)$$

The second feature, *DepCat*, uses dependency patterns to categorize the sentences rather than ranking them. It can be used independently from other features to categorize each sentence by one of the categories described in Section 2.3.1. To do this, we obtain the relational patterns for the current sentence, check whether for each such pattern whether it is included in the *DpM*, and, if so, we add to the sentence the category the pattern was manually associated with. It should be noted that a sentence can have more than one category. This can occur, for instance, if the sentence contains information about when *something* was built and at the same time where it is located. It is also important to mention that assigning sentences categories does not change the order in the ranked list.

We use *DepCat* to generate an automated summary by first including sentences containing the category "type", then "year" and so on until the

[3] In Aker and Gaizauskas (2009) this feature is called *modelSimilarity*.

summary length is violated. The sentences are selected according to the order in which they occur in the ranked list. From each of the first three categories ("type", "year" and "location") we take a single sentence to avoid redundancy. The same is applied to the final two categories ("surrounding" and "visiting"). Then, if length limit is not violated, we fill the summary with sentences from the "background" category until the word limit of 200 words is reached. Here the number of added sentences is not limited. Finally, we order the sentences by first adding the sentences from the first three categories to the summary, then the "background" related sentences and finally the last two sentences from the "surrounding" and "visiting" categories. However, in cases where we have not reached the summary word limit because of uncovered categories, i.e. there were not, for instance, sentences about "location", we add to the end of the summary the next top sentence from the ranked list that was not taken.

3.2 Sentence Selection

To compute the final score for each sentence Aker and Gaizauskas (2009) use a linear function with weighted features:

$$S_{score} = (\sum_{i=1}^{n} feature_i * weight_i) \qquad (2)$$

We use the same approach, but whereas the feature weights they use are experimentally set rather than learned, we learn the weights using linear regression instead. We used $\frac{2}{3}$ of the 310 images from our image set (see Section 4.1) to train the weights. The image descriptions from this data set are used as model summaries.

Our training data contains for each image a set of image descriptions taken from the *VirtualTourist* travel community web-site [4]. From this web-site we took all existing image descriptions about a particular image or object. Note that some of these descriptions about a particular object were used to derive the model summaries for that object (see Section 4.1). Assuming that model summaries contain the most relevant sentences about an object we perform ROUGE comparisons between the sentences in all the image descriptions and the model summaries, i.e. we pair each sentence from all image descriptions about a particular place with every sentence from all the model summaries for that particular object. Sentences which are exactly the same or have common parts will score higher in ROUGE than sentences which do not have anything in common. In this way, we have for each sentence from all existing image descriptions about an object a ROUGE score[5] indicating its relevance. We also ran the summarizer for each of these sentences to compute the values for the different features. This gives information about each feature's value for each sentence. Then the ROUGE scores and feature score values for every sentence were input to the linear regression algorithm to train the weights.

Given the weights, Equation 2 is used to compute the final score for each sentence. The final sentence scores are used to sort the sentences in the descending order. This sorted list is then used by the summarizer to generate the final summary as described in Aker and Gaizauskas (2009).

4 Evaluation

To evaluate our approach we used two different assessment methods: ROUGE (Lin, 2004) and manual readability. In the following we first describe the data sets used in each of these evaluations, and then we present the results of each assessment.

4.1 Data sets

For evaluation we use the image collection described in Aker and Gaizauskas (2010). The image collection contains 310 different images with manually assigned toponyms. The images cover 60 of the 107 object types identified from Wikipedia (see Table 2). For each image there are up to four short descriptions or model summaries. The model summaries were created manually based on image descriptions taken from *VirtualTourist* and contain a minimum of 190 and a maximum of 210 words. An example model summary about the *Eiffel Tower* is shown in Table 4. $\frac{2}{3}$ of this image collection was used to train the weights and the remaining $\frac{1}{3}$ (105 images) for evaluation.

To generate automatic captions for the images we automatically retrieved the top 30 related web-documents for each image using the Yahoo! search engine and the toponym associated with the image as a query. The text from these documents was extracted using an HTML parser and passed to the summarizer. The set of documents we used to generate our summaries excluded any *VirtualTourist* related sites, as these were used to generate

[4] www.virtualtourist.com

[5] We used ROUGE 1.

Table 4: Model, Wikipedia baseline and starterSimilarity+LMSim+DepCat summary for Eiffel Tower.

Model Summary	Wikipedia baseline summary	starterSimilarity+LMSim+DepCat summary
The Eiffel Tower is the most famous place in Paris. It is made of 15,000 pieces fitted together by 2,500,000 rivets. It's of 324 m (1070 ft) high structure and weighs about 7,000 tones. This world famous landmark was built in 1889 and was named after its designer, engineer Gustave Alexandre Eiffel. It is now one of the world's biggest tourist places which is visited by around 6,5 million people yearly. There are three levels to visit: Stages 1 and 2 which can be reached by either taking the steps (680 stairs) or the lift, which also has a restaurant "Altitude 95" and a Souvenir shop on the first floor. The second floor also has a restaurant "Jules Verne". Stage 3, which is at the top of the tower can only be reached by using the lift. But there were times in the history when Tour Eiffel was not at all popular, when the Parisians thought it looked ugly and wanted to pull it down. The Eiffel Tower can be reached by using the Mtro through Trocadro, Ecole Militaire, or Bir-Hakeim stops. The address is: Champ de Mars-Tour Eiffel.	The Eiffel Tower (French: Tour Eiffel, [tur efel]) is a 19th century iron lattice tower located on the Champ de Mars in Paris that has become both a global icon of France and one of the most recognizable structures in the world. The Eiffel Tower, which is the tallest building in Paris, is the single most visited paid monument in the world; millions of people ascend it every year. Named after its designer, engineer Gustave Eiffel, the tower was built as the entrance arch for the 1889 World's Fair. The tower stands at 324 m (1,063 ft) tall, about the same height as an 81-story building. It was the tallest structure in the world from its completion until 1930, when it was eclipsed by the Chrysler Building in New York City. Not including broadcast antennas, it is the second-tallest structure in France, behind the Millau Viaduct, completed in 2004. The tower has three levels for visitors. Tickets can be purchased to ascend either on stairs or lifts to the first and second levels.	The Eiffel Tower, which is the tallest building in Paris, is the single most visited paid monument in the world; millions of people ascend it every year. The tower is located on the Left Bank of the Seine River, at the northwestern extreme of the Parc du Champ de Mars, a park in front of the Ecole Militaire that used to be a military parade ground. The tower was met with much criticism from the public when it was built, with many calling it an eyesore. Counting from the ground, there are 347 steps to the first level, 674 steps to the second level, and 1,710 steps to the small platform on the top of the tower. Although it was the world's tallest structure when completed in 1889, the Eiffel Tower has since lost its standing both as the tallest lattice tower and as the tallest structure in France. The tower has two restaurants: Altitude 95, on the first floor 311ft (95m) above sea level; and the Jules Verne, an expensive gastronomical restaurant on the second floor, with a private lift.

Table 5: ROUGE scores for each single feature and Wikipedia baseline.

Recall	centroidSimilarity	sentencePosition	querySimilarity	starterSimilarity	LMSim	DpMSim***	Wiki
R2	.0734	**.066**	.0774	.0869	.0895	**.093**	.097
RSU4	.12	**.11**	.12	.137	.142	**.145**	.14

the model summaries.

4.2 ROUGE assessment

In the first assessment we compared the automatically generated summaries against model summaries written by humans using ROUGE (Lin, 2004). Following the Document Understanding Conference (DUC) evaluation standards we used ROUGE 2 (*R2*) and ROUGE SU4 (*RSU4*) as evaluation metrics (Dang, 2006) . ROUGE 2 gives recall scores for bi-gram overlap between the automatically generated summaries and the reference ones. ROUGE SU4 allows bi-grams to be composed of non-contiguous words, with a maximum of four words between the bi-grams.

As baselines for evaluation we used two different summary types. Firstly, we generated summaries for each image using the top-ranked non Wikipedia document retrieved in the Yahoo! search results for the given toponyms. From this document we create a baseline summary by selecting sentences from the beginning until the summary reaches a length of 200 words. As a second baseline we use the Wikipedia article for a given toponym from which we again select sentences from the beginning until the summary length limit is reached.

First, we compared the baseline summaries against the VirtualTourist model summaries. The comparison shows that the Wikipedia baseline ROUGE scores (R2 .097***, RSU4 .14***) are significantly higher than the first document ones (R2 0.042, RSU4 .079) [6]. Thus, we will focus on the Wikipedia baseline summaries to draw conclusions about our automatic summaries. Table 4 shows the Wikipedia baseline summary about the *Eiffel Tower*.

Secondly, we separately ran the summarizer over the top ten documents for each single feature and compared the automated summaries against the model ones. The results of this comparison are shown in Table 5.

Table 5 shows that the dependency model feature (*DpMSim*) contributes most to the summary quality according to the ROUGE metrics. It is also significantly better than all other feature scores except the *LMSim* feature. Compared to *LMSim* ROUGE scores the *DpMSim* feature offers only a moderate improvement. The same moderate improvement we can see between the *DpMSim* RSU4 and the *Wiki* RSU4. The lowest ROUGE scores are obtained if only sentence position (*sentecePosition*) is used.

To see how the ROUGE scores change when features are combined with each other we performed different combinations of the features, ran the summarizer for each combination and compared the automated summaries against the model ones. In the different combinations we

[6] To assess the statistical significance of ROUGE score differences between multiple summarization results we performed a pairwise Wilcoxon signed-rank test. We use the following conventions for indicating significance level in the tables: *** = $p < .0001$, ** = $p < .001$, * = $p < .05$ and no star indicates non-significance.

Table 6: ROUGE scores of feature combinations which score moderately or significantly higher than dependency pattern model (*DpMSim*) feature and Wikipedia baseline.

Recall	starterSimilarity + LMSim	starterSimilarity + LMSim + DepCat***	DpmSim	Wiki
R2	.095	**.102**	.093	.097
RSU4	.145	**.155**	.145	.14

also included the dependency pattern categorization (*DepCat*) feature explained in Section 3.1. Table 6 shows the results of feature combinations which score moderately or significantly higher than the dependency pattern model (*DpMSim*) feature score shown in Table 5.

The results showed that combining *DpMSim* with other features did not lead to higher ROUGE scores than those produced by that feature alone.

The summaries categorized by dependency patterns (*starterSimilarity+LMSim+DepCat*) achieve significantly higher ROUGE scores than the Wikipedia baseline. For both ROUGE R2 and ROUGE SU4 the significance is at level $p < .0001$. Table 4 shows a summary about the *Eiffel Tower* obtained using this *starterSimilarity+LMSim+DepCat* feature. Table 5 also shows the ROUGE scores of the feature combination *starterSimilarity* and *LMSim* used without the dependency categorization (*DepCat*) feature. It can be seen that this combination without the dependency patterns lead to lower ROUGE scores in ROUGE 2 and only moderate improvement in ROUGE SU4 if compared with Wikipedia baseline ROUGE scores.

4.3 Readability assessment

We also evaluated our summaries using a readability assessment as in DUC and TAC. DUC and TAC manually assess the quality of automatically generated summaries by asking human subjects to score each summary using five criteria – *grammaticality, redundancy, clarity, focus* and *structure* criteria. Each criterion is scored on a five point scale with high scores indicating a better result (Dang, 2005).

For this evaluation we used the same 105 images as in the ROUGE evaluation. As the ROUGE evaluation showed that the dependency pattern categorization (*DepCat*) renders the best results when used in feature combination *starterSimilarity + LMSim + DepCat*, we further investigated the contribution of dependency pattern categorization by performing a readability assessment on summaries generated using this feature combination. For comparison we also evaluated summaries which were not structured by dependency patterns (*starterSimilarity + LMSim*) and also the Wikipedia baseline summaries.

We asked four people to assess the summaries. Each person was shown all 315 summaries (105 from each summary type) in a random way and was asked to assess them according to the DUC and TAC manual assessment scheme. The results are shown in Table 7.

We see from Table 7 that using dependency patterns to categorize the sentences and produce a structured summary helps to obtain better readable summaries. Looking at the 5 and 4 scores the table shows that the dependency pattern categorized summaries (*SLMD*) have better clarity (85% of the summaries), are more coherent (74% of the summaries), contain less redundant information (83% of the summaries) and have better grammar (92% of the summaries) than the ones without dependency categorization (80%, 70%, 60%, 84%).

The scores of our automated summaries were better than the Wikipedia baseline summaries in the *grammar* feature. However, in other features the Wikipedia baseline summaries obtained better scores than our automated summaries. This comparison show that there is a gap to fill in order to obtain better readable summaries.

5 Related Work

Our approach has an advantage over related work in automatic image captioning in that it requires only GPS information associated with the image in order to generate captions. Other attempts towards automatic generation of image captions generate captions based on the immediate textual context of the image with or without consideration of image related features such as colour, shape or texture (Deschacht and Moens, 2007; Mori et al., 2000; Barnard and Forsyth, 2001; Duygulu et al., 2002; Barnard et al., 2003; Pan et al., 2004; Feng and Lapata, 2008; Satoh et al., 1999; Berg et al., 2005). However, Marsch & White (2003) argue that the content of an image and its immediate text have little semantic agreement and this can, according to Purves et al. (2008), be misleading to image retrieval. Furthermore, these approaches assume that the image has been obtained from a document. In cases where there is no document associated with the image, which is the scenario we are principally concerned with, these techniques are not applicable.

Table 7: Readability evaluation results: Each cell shows the percentage of summaries scoring the ranking score heading the column for each criterion in the row as produced by the summary method indicated by the subcolumn heading – Wikipedia baseline (W), starterSimilarity + LMSim (SLM) and starterSimilarity + LMSim + DepCat (SLMD). The numbers indicate the percentage values averaged over the four people.

	5			4			3			2			1		
Criterion	W	SLM	SLMD	W	SLM	SLMD	W	SLM	SLMD	W	SLM	SLMD	W	SLM	SLMD
clarity	72.6	50.5	53.6	21.7	30.0	31.4	1.2	6.7	5.7	4.0	10.2	6.0	0.5	2.6	3.3
focus	72.1	49.3	51.2	20.5	26.0	25.2	3.8	10.0	10.7	3.3	10.0	10.5	0.2	4.8	2.4
coherence	67.1	39.0	48.3	23.6	31.4	26.9	4.8	12.4	11.9	3.3	10.2	9.8	1.2	6.9	3.1
redundancy	69.8	42.9	55.0	21.7	17.4	28.8	2.4	4.5	4.3	5.0	27.1	8.8	1.2	8.1	3.1
grammar	48.6	55.7	62.9	32.9	29.0	30.0	5.0	3.1	1.9	11.7	12.1	5.2	1.9	0	0

Dependency patterns have been exploited in various language processing applications. In information extraction, for instance, dependency patterns have been used to extract relevant information from text resources (Yangarber et al., 2000; Sudo et al., 2001; Culotta and Sorensen, 2004; Stevenson and Greenwood, 2005; Bunescu and Mooney, 2005; Stevenson and Greenwood, 2009). However, dependency patterns have not been used extensively in summarization tasks. We are aware only of the work described in Nobata et al. (2002) who used dependency patterns in combination with other features to generate extracts in a single document summarization task. The authors found that when learning weights in a simple feature weigthing scheme, the weight assigned to dependency patterns was lower than that assigned to other features. The small contribution of the dependency patterns may have been due to the small number of documents they used to derive their dependency patterns – they gathered dependency patterns from only ten domain specific documents which are unlikely to be sufficient to capture repeated features in a domain.

6 Discussion and Conclusion

We have proposed a method by which dependency patterns extracted from corpora of descriptions of instances of particular object types can be used in a multi-document summarizer to automatically generate image descriptions. Our evaluations show that such an approach yields summaries which score more highly than an approach which uses a simpler representation of an object type model in the form of a n-gram language model.

When used as the sole feature for sentence ranking, dependency pattern models (*DpMSim*) produced summaries with higher ROUGE scores than those obtained using the features reported in Aker and Gaizauskas (2009). These dependency pattern models also achieved a modest improvement over Wikipedia baseline ROUGE SU4. Furthermore, we showed that using dependency patterns in combination with features reported in Aker and Gaizauskas to produce a structured summary led to significantly better results than Wikipedia baseline summaries as assessed by ROUGE. However, human assessed readability showed that there is still scope for improvement.

These results indicate that dependency patterns are worth investigating for object focused automated summarization tasks. Such investigations should in particular concentrate on how dependency patterns can be used to structure information within the summary, as our best results were achieved when dependency patterns were used for this purpose.

There are a number of avenues to pursue in future work. One is to explore how dependency patterns could be used to produce generative summaries and/or perform sentence trimming. Another is to investigate how dependency patterns might be automatically clustered into groups expressing similar or related facts, rather than relying on manual categorization of dependency patterns into categories such as “type”, “year”, etc. as was done here. Evaluation should be extended to investigate the utility of the automatically generated image descriptions for image retrieval. Finally, we also plan to analyze automated ways for learning information structures (e.g. what is the flow of facts to describe a location) from existing image descriptions to produce better summaries.

7 Acknowlegment

The research reported was funded by the TRIPOD project supported by the European Commission under the contract No. 045335. We would like to thank Emina Kurtic, Mesude Bicak, Edina Kurtic and Olga Nesic for participating in our manual evaluation. We also would like to thank Trevor Cohn and Mark Hepple for discussions and comments.

References

A. Aker and R. Gaizauskas. 2009. Summary Generation for Toponym-Referenced Images using Object

Type Language Models. *International Conference on Recent Advances in Natural Language Processing (RANLP),2009*.

A. Aker and R. Gaizauskas. 2010. Model Summaries for Location-related Images. In *Proc. of the LREC-2010 Conference*.

K. Barnard and D. Forsyth. 2001. Learning the semantics of words and pictures. In *International Conference on Computer Vision*, volume 2, pages 408–415. Vancouver: IEEE.

K. Barnard, P. Duygulu, D. Forsyth, N. de Freitas, D.M. Blei, and M.I. Jordan. 2003. Matching words and pictures. *The Journal of Machine Learning Research*, 3:1107–1135.

T.L. Berg, A.C. Berg, J. Edwards, and DA Forsyth. 2005. Whos in the Picture? In *Advances in Neural Information Processing Systems 17: Proc. Of The 2004 Conference*. MIT Press.

R.C. Bunescu and R.J. Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 724–731. Association for Computational Linguistics Morristown, NJ, USA.

A. Culotta and J. Sorensen. 2004. Dependency Tree Kernels for Relation Extraction. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 423–429, Barcelona, Spain, July.

H.T. Dang. 2005. Overview of DUC 2005. *DUC 05 Workshop at HLT/EMNLP*.

H.T. Dang. 2006. Overview of DUC 2006. *National Institute of Standards and Technology*.

K. Deschacht and M.F. Moens. 2007. Text Analysis for Automatic Image Annotation. *Proc. of the 45th Annual Meeting of the Association for Computational Linguistics. East Stroudsburg: ACL.*

P. Duygulu, K. Barnard, JFG de Freitas, and D.A. Forsyth. 2002. Object Recognition as Machine Translation: Learning a Lexicon for a Fixed Image Vocabulary. *In Seventh European Conference on Computer Vision (ECCV)*, 4:97–112.

X. Fan, A. Aker, M. Tomko, P. Smart, M Sanderson, and R. Gaizauskas. 2010. Automatic Image Captioning From the Web For GPS Photographs. In *Proc. of the 11th ACM SIGMM International Conference on Multimedia Information Retrieval, National Constitution Center, Philadelphia, Pennsylvania*.

Y. Feng and M. Lapata. 2008. Automatic Image Annotation Using Auxiliary Text Information. *Proc. of Association for Computational Linguistics (ACL) 2008, Columbus, Ohio, USA*.

C.Y. Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. *Proc. of the Workshop on Text Summarization Branches Out (WAS 2004)*, pages 25–26.

E.E. Marsh and M.D. White. 2003. A taxonomy of relationships between images and text. *Journal of Documentation*, 59:647–672.

Y. Mori, H. Takahashi, and R. Oka. 2000. Automatic word assignment to images based on image division and vector quantization. In *Proc. of RIAO 2000: Content-Based Multimedia Information Access*.

C. Nobata, S. Sekine, H. Isahara, and R. Grishman. 2002. Summarization system integrated with named entity tagging and ie pattern discovery. In *Proc. of the LREC-2002 Conference*, pages 1742–1745.

J.Y. Pan, H.J. Yang, P. Duygulu, and C. Faloutsos. 2004. Automatic image captioning. In *Multimedia and Expo, 2004. ICME'04. IEEE International Conference on*, volume 3.

RS Purves, A. Edwardes, and M. Sanderson. 2008. Describing the where–improving image annotation and search through geography. *1st Intl. Workshop on Metadata Mining for Image Understanding, Funchal, Madeira-Portugal*.

S. Satoh, Y. Nakamura, and T. Kanade. 1999. Name-It: naming and detecting faces in news videos. *Multimedia, IEEE*, 6(1):22–35.

F. Song and W.B. Croft. 1999. A general language model for information retrieval. In *Proc. of the eighth international conference on Information and knowledge management*, pages 316–321. ACM New York, NY, USA.

M. Stevenson and M.A. Greenwood. 2005. A semantic approach to IE pattern induction. In *Proc. of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 379–386. Association for Computational Linguistics Morristown, NJ, USA.

M. Stevenson and M. Greenwood. 2009. Dependency Pattern Models for Information Extraction. *Research on Language and Computation*, 7(1):13–39.

K. Sudo, S. Sekine, and R. Grishman. 2001. Automatic pattern acquisition for Japanese information extraction. In *Proc. of the first international conference on Human language technology research*, page 7. Association for Computational Linguistics.

R. Yangarber, R. Grishman, P. Tapanainen, and S. Huttunen. 2000. Automatic acquisition of domain knowledge for information extraction. In *Proc. of the 18th International Conference on Computational Linguistics (COLING 2000)*, pages 940–946. Saarbriicken, Germany, August.