ANVIL: a System for the Retrieval of Captioned Images using NLP Techniques

Tony Rose, David Elworthy, Aaron Kotcheff and Amanda Clare

Canon Research Centre Europe
Guildford, Surrey

Petros Tsonis

Dept. of Computing Science, University of Glasgow
Scotland

Abstract

ANVIL is a system designed for the retrieval of images annotated with short captions. It uses NLP techniques to extract dependency structures from captions and queries, and then applies a robust matching algorithm to recursively explore and compare them. There are currently two main interfaces to ANVIL: a list-based display and a 2D spatial layout that allows users to interact with and navigate between similar images.

ANVIL was designed to operate as part of a publicly accessible, WWW-based image retrieval server. Consequently, product-level engineering standards were required. This paper examines both the research aspects of the system and also looks at some of the design and evaluation issues.

1 Introduction

A fundamental aim of many Information Retrieval (IR) systems is to match a user's statement of their information need with the contents of a document collection. Since queries and documents are often predominantly textual, it seems reasonable to suggest that interpretation of their underlying linguistic structure would be beneficial. However, in practice, this has rarely proved to be the case [1].

One notable exception is the Intermezzo system [2], in which NLP techniques were applied to the retrieval of images that had been annotated with short captions. This system returned a precision of around 90% for 50 queries on a 500,000-image database. This result would tend to indicate that the combination of a short caption (as opposed to a larger document) with some control over its content (as opposed to minimal editorial control over large document collections) may facilitate the application of specific NLP techniques to improve the retrieval process. This paper, and the ANVIL image retrieval system it describes, constitutes a further investigation of this hypothesis.

1.1 Design Goals

ANVIL is designed for the retrieval of captioned images, using fast and accurate natural language techniques. As with Intermezzo, we use a database of images with phrasal captions, their length ranging from 1 to 20 words (mean = ~9 words). Some examples are as follows:

```
Golden columns reaching up towards a deep blue sky
Blue surface covered in stippled yellow paint
Big Ben, London
```

However, unlike Intermezzo, rather than trying to identify compound terms to use in the indexing process, we apply a lightweight parse to the captions and queries and then attempt to measure their similarity by comparing the analysed structures. ANVIL uses a finite-state parser that extracts dependency relations that are then recursively compared and assigned a score for the match based on their similarity. This process is combined with keyword matching to produce an overall score upon which the output ranking is based.

ANVIL was designed to operate primarily as a WWW-based image retrieval server, accessible via a number of user interface clients. However, a standalone variant has also been developed, accessible as a command line executable program. The results of an example query to this version is shown below (with some elementary reformatting for clarity):

```
Query = 'camera with a lens'
```

```
3 results:

SCORE   CAPTION
=====   =======
1.0     black SLR camera, with zoom lens, on a white surface.

0.5     Canon camera, magnifying lens and fashion magazine on grey ridge surface.

0.1     an astronaut floating within a space craft, showing the on-board cameras.
```

The implementation work was done in collaboration with a Japanese development team within another part of Canon. Consequently, ANVIL had to be designed to support both Japanese and English language input. This required the language-specific components to be designed with clear interfaces to rest of the system, and to be modular, to facilitate replacement or modification should support for other languages become necessary.

2. Overview of ANVIL

The overall design of ANVIL is shown in Figure 1. There are two paths through ANVIL, one for each type of input (caption or query). In the case of captions, the pre-processor reads the input via a character stream and returns a set of 'analyses', each of which consists of a set of tokens augmented with lexical information. The analyses are then parsed to identify the phrase bracketing and modification relationships between the tokens. Finally, the caption, analyses and parse information are then stored in the database.

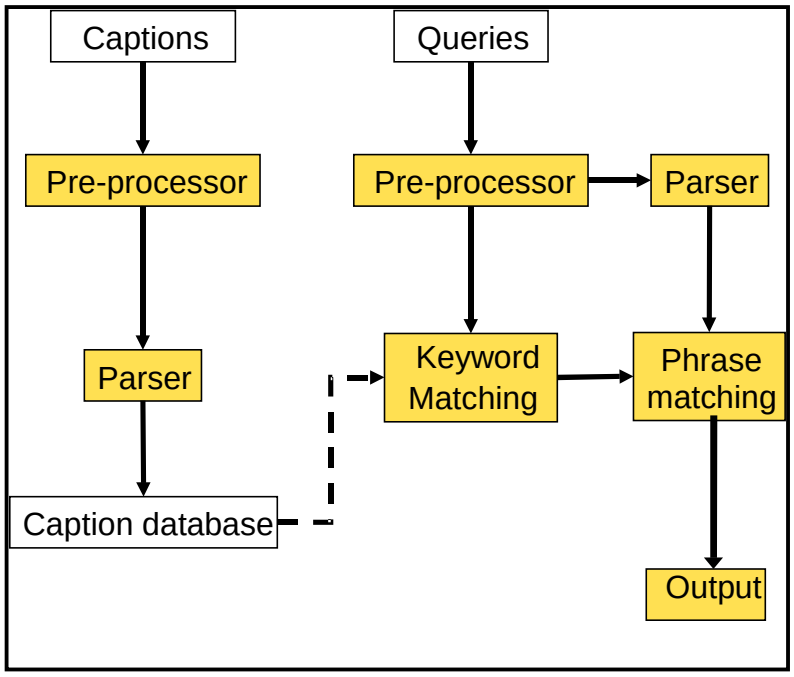

Figure 1: The overall ANVIL architecture

In the case of a query, the processing is somewhat more complex. As with captions, the input is first pre-processed and parsed. However, following pre-processing, the content words are now identified and used to perform a lookup of the caption database to identify a set of initial candidates for keyword matching. A number of standard keyword-based IR techniques are applied in this process, which provides the first set of match scores to be used in ranking the output. This is followed by phrase matching, in which the dependency structures of the query and the candidate captions are compared. This produces the second set of match scores that are used in ranking the output. Finally, the two sets of match scores are combined in the output component. Output is provided in the form of an XML-like data stream suitable for interpretation and display by a range of user interface clients (e.g. those of Section 6).

2.1 The Pre-processor

The purpose of the pre-processor is to perform the orthographic and lexical processing of the input. Consequently, most of the language-dependent parts of ANVIL are contained within this component. For Japanese, this involves a process known as *bunsetsu-splitting*, in which the input character stream is split into content-bearing meaning elements by looking for particles and inflectional endings. The design of the English pre-processor is shown in Figure 2.

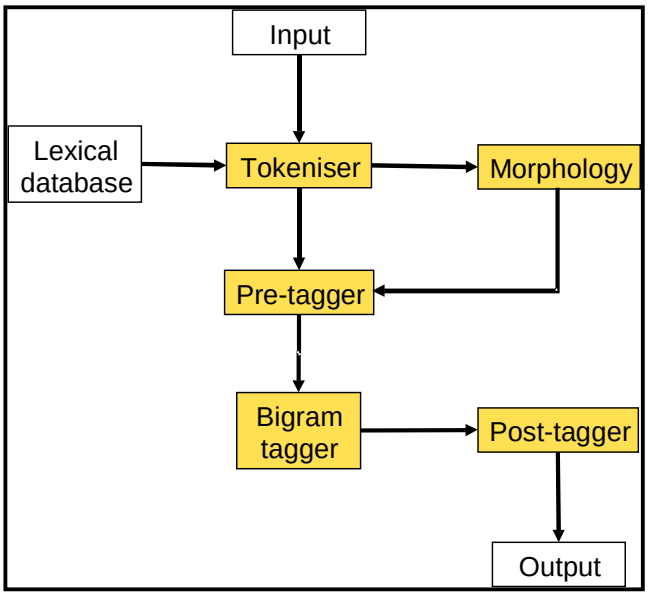

Figure 2: The English Pre-processor

The pre-processor reads the input and passes it first to the tokeniser, which splits the character stream into individual words. In many cases, these will be simple words, separated by white spaces, but in others the tokeniser will need to reliably identify special patterns such as numbers, dates and multi-word units (e.g. "*Houses of Parliament*" and "*in so far as*").

The tokeniser looks each word up directly in a lexicon of 170,000 fully inflected forms. The lexicon stores a variety of data for each word; the most important of which are the tag data and the word sense information. The tag data consists of part-of-speech (POS) tags and their *a-priori* probabilities, which are required in subsequent processing by the bigram tagger. The word senses are integers representing the meaning of each word, using a system of identifiers extracted from WordNet [3]. This system is designed such that synonyms are grouped into sets with a common identifier (known as a "synsetID"). The synsetIDs are augmented with a score to indicate their relative importance.

Even though the lexicon is relatively large, it is inevitable that some lookup operations will fail. In such cases, the morphological analyser attempts to identify the general syntactic category (noun, verb, adjective, adverb or unknown) of the token. It does this by comparing the token with a pre-defined set of affixes and associated syntactic categories. The accuracy of this operation is improved by applying various heuristics, e.g. looking for longest match first, checking that the resultant stem contains at least one vowel, and so on.

The output from the tokeniser is a lattice of analyses corresponding to the various ways in which the input can be tokenised. This lattice is then flattened into a sequence of separate tokenisations for input to the tagger. The tagger is a standard HMM-based, bigram tagger that was trained offline, using standard techniques, on text from the British National Corpus (http://info.ox.ac.uk/bnc). In most cases, it assigns a single POS tag to each word. However, some ambiguous words may receive more than one tag, resulting in the need for a further lattice-like structure as the output. This lattice is then flattened in the post-tagger, using an algorithm similar to that of the pre-tagger.

2.2. Lexical resources

The acquisition and maintenance of a lexicon of 170,000 fully inflected forms is a considerable challenge, requiring the painstaking processing of information from a variety of lexical resources. The primary source of lexical information was a wordlist extracted from the BNC for word types with a frequency of 20 or greater. This was

augmented by further data from WordNet, in particular the synsetIDs, US English spellings and many multi-word units.

WordNet is also used for the calculation of the related terms provided by the query expansion process. These related terms are identified using a process known as *semantic broadening*, which is based on a procedure described in [4]. The basic idea is that for each synsetID, we traverse the WordNet 'ISA' hierarchy to find more specific terms (*hyponyms*) and more general terms (*hypernyms*). Similarity is measured by considering both the distance between nodes in the hierarchy and their information content (as measured using frequencies extracted from the BNC and WordNet's semantic concordance). Once these related words have been identified (offline), they are stored in the lexicon such that a simple lookup is all that is required at run time to expand the query (see Section 3).

3. Keyword matching

Keyword matching in ANVIL uses a number of standard IR techniques to compare queries with captions. However, there are a number of important differences. Firstly, despite the name, the keywords are actually the content words from the caption (or query), as opposed to metadata keywords as typically used in the annotation of many commercial stock photography collections. (Although ANVIL can also handle this sort of keyword input, we will not discuss such processing here.) These keywords may be single words or compound terms. Secondly, the unit of comparison is the synsetID, so some elementary query expansion is taking place implicitly, since synonyms will share the same synsetID. In addition, the query is further expanded using the semantic broadening process described in Section 2.3.

The process begins with a database lookup on the content terms in the query to retrieve a list of candidate matching captions. ANVIL can be configured to limit the size of this list to a fixed value as a way of controlling the overall retrieval time (at the risk of missing some captions). ANVIL will then apply one of four keyword matching methods (depending on the configuration): vector cosine, two variants of the "run 6" algorithm described in [5] or a simplified version of the BM25 probabilistic formula [6]. In practice, each of these methods produces similar results, although the algorithm described in [5] gives slightly better accuracy on our test data and runs slightly faster. A further configuration parameter may then be used to specify a threshold below which low-scoring captions will be discarded.

4. The Parser

ANVIL uses a cascaded, finite-state parser, which means that the grammar is divided into a number of levels ('cascades') and the rules are written in the form of regular expressions ('finite-state machines'). Input is in the form of a sequence of tokens and their associated POS tags, and the output consists of phrase bracketing and the modification relationships between the tokens. The rules are written in terms of lexical parts of speech, phrasal tags and specific words (to handle special cases).

Each grammar rule specifies a phrasal tag that will be constructed if the right hand side of the rule matches the input. For example:

```
np -> det? adj* noun
```

specifies that a noun phrase (`np`) will be constructed if the input consists of an optional determiner, followed by any number of adjectives and a noun. This process provides bracketing information, but for modification relationships a system of variables is required. We therefore augment our notation with further variables, which are represented as elements shown after a colon in the notation. For example, the above rule could become:

```
np -> det? adj:mod{head}* noun:head
```

The variables are instantiated whenever the body of the rule matches the input. They can be indexed on another variable, using the '`{}`' notation. So in the above rule, the list of adjectives in the input is stored in the variable `mod`, which is indexed on the variable `head`. Alternatively, variables can be unindexed. So in the above rule, the input noun would be stored in the variable `head`. For example, if the input was "*red sunset*", then "*sunset*" would be stored in `head`, and "*red*" would be stored in `mod{sunset}`. The variables allow us to record relationships between words, which when combined with the cascaded design of the grammar rules, allows us to represent nested dependency structures that can be used highly effectively in subsequent matching operations. The grammar rules are compiled offline into finite-state machines and read into ANVIL at run time. This allows us to

iteratively refine the grammar separately from the development of the ANVIL code itself.

5. Phrase matching

The goal of phrase matching is to increase the accuracy of retrieval in ANVIL. It does this by matching queries with captions, using the dependency structures produced by the parser. Phrase matching starts with the most important words and then descends through the dependency structure, either matching other words or skipping over them.

5.1 Phrase Matching Rules

The process is guided by a set of rules that specify the types of structures that are allowed to match. In addition, the rules are augmented by scores to indicate the strength of various matches and their relative importance. The rules may also contain fixed expressions to match against certain special cases in the input, e.g. negation. A small set of example rules is shown below:

```
head_rule {
  head              =     head   1.0          => mod_rule 0.7;
  head              =     mod[]  0.5          => mod_rule 0.7;
  mod[]             ?            0.3          => Done 1.0;
}

mod_rule {
  mod[]             =     mod[] 1.0           => mod_rule 1.0;

  phead:prep[]      =     mod[] 1.0           => mod_rule 1.0;
  mod[]             =     phead:prep[] 1.0    => mod_rule 1.0;
  phead:prep[]      =     phead:prep[] 1.0    => mod_rule 1.0;

  amod[]            =     amod[] 1.0          => Done 1.0;
  amod              =     'not'  0.0          => Done 0.0;
  'not'             =     amod   0.0          => Done 0.0;
}
```

The rules are composed of two main elements: a comparison and a continuation (we shall ignore the numerical elements for now). The comparison part represents a path through the dependency structure, and is specified by the items on the left hand side of the arrow (=>). The continuation part is specified by the item on the right hand side of the arrow, and this represents the rule that will be invoked if the comparison produces a match.

The rules are divided into levels. In the above example we have the two levels; the first being the `head_rule`, which is where processing starts. The `head_rule` above specifies that the heads of the query and caption should first be compared. If they match, then the `mod_rule` is invoked. Alternatively, if the head of the caption matches any of the query modifiers, then again the `mod_rule` is invoked. The third comparison in the `head_rule` (containing the '`?`' symbol) allows us to 'skip over' any other words which do not have an effect on the match.

Within the `mod_rule` (or any other continuation rule), the [] symbol means that each word that matched at the previous level (i.e. in this case the `head_rule`) is inserted. This allows the current rule to access variables instantiated by a previous rule (e.g. if the heads had matched in the `head_rule` then the first part of the `mod_rule` allows us to compare their modifiers). Rules may also contain literal tokens (e.g. '`not`') which allows us to identify cases of negation by checking whether an `amod` (adverbial modifier) matches the literal word "*not*". The process of phrase matching continues down each branch of the dependency structure until the special continuation `Done` is reached.

5.2 The Scoring System

The numerical parts of the rules allow us to provide a measure of how well the caption and query match. There are two main elements to this: the *term factor*, which appears immediately after the comparison; and the *down factor*, which appears immediately after the continuation. The first of these can be thought of as a measure of match strength, and the latter of its importance. In the first part of the `head_rule` above the term factor is 1.0 and the down factor 0.7.

As the matching process proceeds, any query words that take part in a match are assigned a *strength score* and a *weight score*. The match strength score is based on the product of the term factor, the similarity with the matching

caption word (calculated using the semantic broadening procedure described in Section 2.3) and the *up-score*, which is the value shown after the special continuation `Done`. The up-score is usually 1.0 (which leaves the match score unchanged), but in the case of negation, it can be used to reduce a match score to zero (hence the up-score of 0.0 in the final two parts of the `mod_rule`).

The weight of each query word is calculated as the product of the down factors involved as each continuation rule is invoked. Once the entire dependency structure has been traversed (i.e. all applicable branches of the rule set have been explored) then the overall match score for the caption is calculated as $\Sigma s_I w_i / s_i$ which returns a value in the range 0-1. This score is then combined with the keyword matching score (using by default the mean of the two scores) to rank the candidate captions returned by ANVIL.

6. User Interfaces

6.1 1-D Design

ANVIL currently has two main user interfaces. The first is a traditional, list-based ('Alta-Vista' style) interface, that presents the matching images and their captions as a ranked list. This is shown in Figure 3. This interface has the benefit that it is familiar, and it is intuitively evident that the highest ranking results will appear at the top of the list, so users can readily scroll down to find further matches. To make the match score more meaningful, a simple 'LED meter' widget was incorporated into the caption display area, so that users can compare match scores without needing to interpret numerical values. However, this type of interface design lacks structure, in that no attempt is made to group the images according to their similarity. For example, the query "*red toy car*" produces images of toys, cars and other related objects, but they are all interleaved in the ranking list.

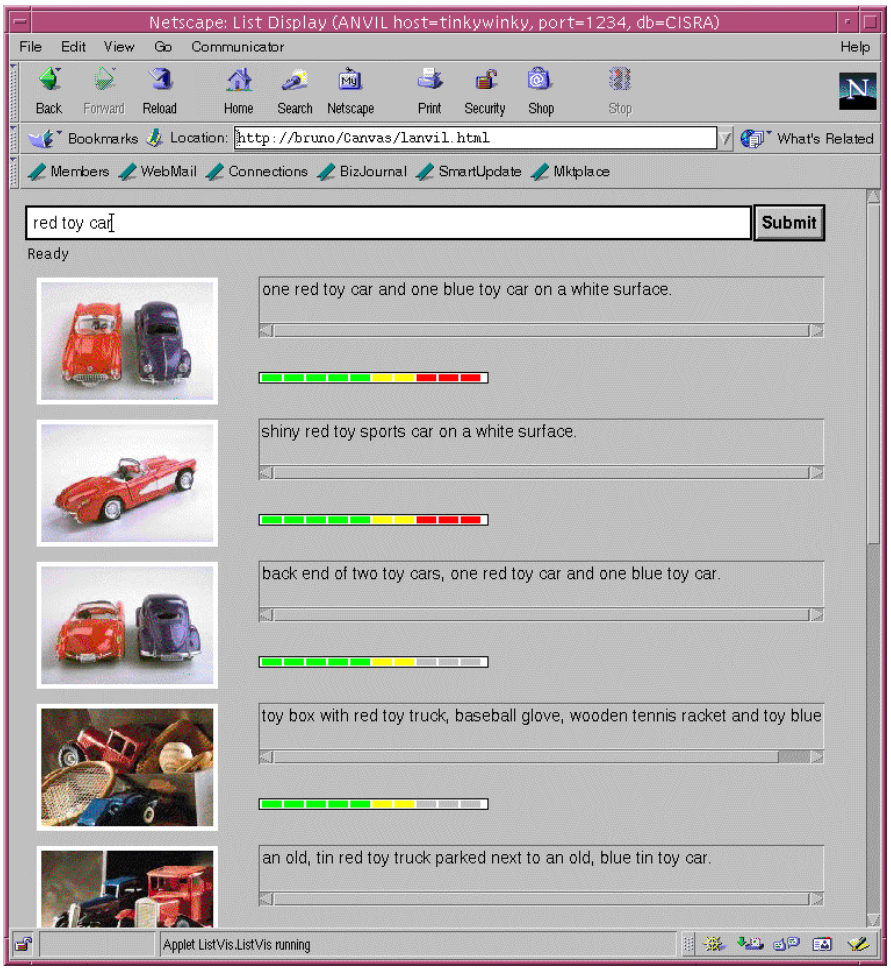

Figure 3: The 1D user interface design

6.2 2-D Design

The second interface to ANVIL attempts to address this shortcoming by utilising a 2-D spatial layout interface (shown in Figure 4), which incorporates a number of novel features. The principle behind this design is that the layout makes clear the best matching image by placing it at the centre of the display (known as the 'focal image'). Other matching images are placed around this at a distance proportional to their similarity to the query. In addition, the other matching images are placed at distances from each other proportional to their mutual similarity. Hence, groups of similar images will tend to form clusters. Similar 2-D designs have been explored by [7], [8] and [9].

Animation is used to indicate to the user how these clusters are formed. On issuing a query, the display clears and the focal image appears, tightly surrounded by the remaining matches. Over the next few seconds the images then re-arrange themselves to find the position of highest mutual stability. The benefit of this is that rather than being confronted with a completed map (in which the nature of the various groupings may or may not be self-evident), the user can actually see the groups forming and hence gain some idea of each image's relative effect on the clustering process. In addition, the use of visual links (i.e. lines) between highly similar or dissimilar images helps to reinforce this effect.

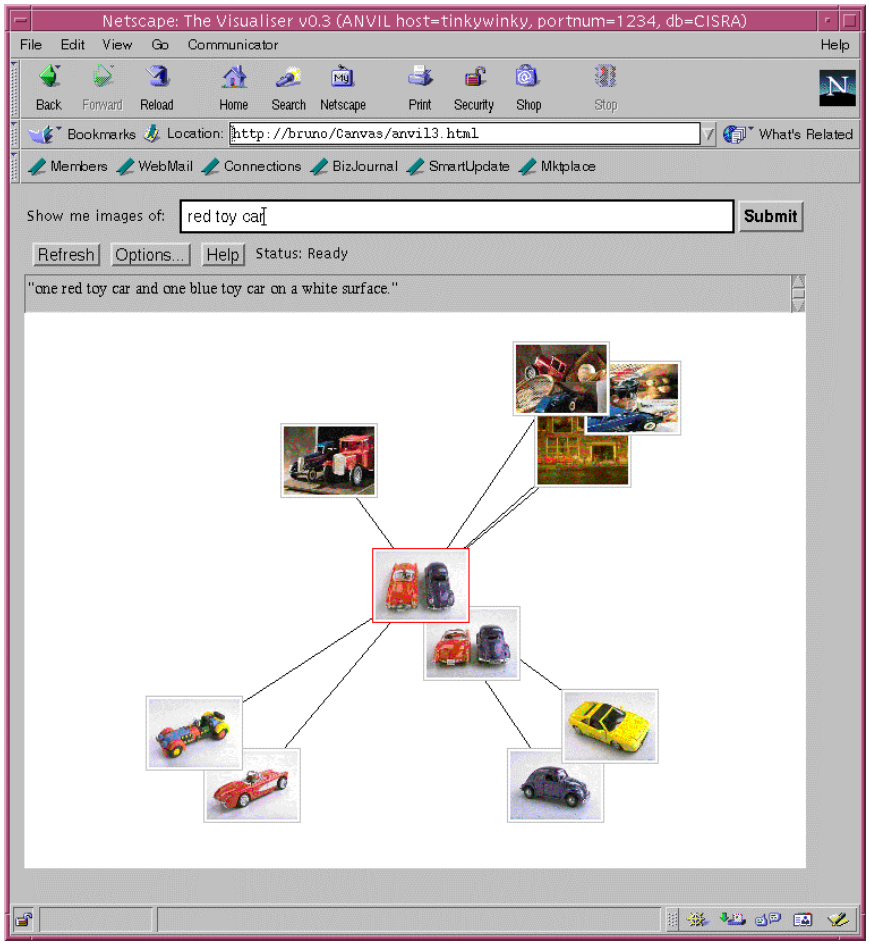

Figure 4: The 2D user interface design

The similarities between the various images in the database are calculated offline, by iteratively applying the ANVIL matching process across all the captions in the database (i.e. treating each one as a separate query). Consequently, a simple database lookup is all that is required at runtime. For large image collections, the offline processing can take a considerable amount of time, so in practice we configure ANVIL to use keyword matching only and set thresholds to limit the number of results.

Evidently, the clustering process can result in some degree of overlap between the images. One way to avoid this is to use a grid layout algorithm so that images are laid out without overlaps but maintaining their relative proximities [10]. Another approach is to allow the user greater control over the manipulation of the images in the

display. The ANVIL 2-D client allows users to magnify and drag images anywhere on the screen, without affecting the linkage topology (i.e. the interconnecting lines). In addition, double clicking on any one thumbnail submits that image as a query-by-example, using its caption as a new query to ANVIL.

7. Evaluation

7.1 Precision & recall

A standard way of evaluating IR systems is to submit a series of queries, look at the documents returned for each one, and then compare these with a known set of relevance judgements (usually assigned in advance by a set of human judgements). This procedure has been common practice in the Text Retrieval community for many years and consequently a variety of standardised test sets are available (e.g. see http://trec.nist.gov). However, for the retrieval of captioned images, standardised test data is somewhat more limited. We therefore developed our own test set, using the following procedure.

First, a set of images was obtained - in this case the 1,700 images in the CISRA PhotoEssentials collection (http://www.photoessentials.com). These were captioned in accordance with a set of guidelines that had been drawn up as a result of an earlier in-house pilot study. Some images were given multiple captions (usually those in which the subject could be described by either a proper name or a general description), e.g. "*Big Ben, London*" and "*tall, ornate clock tower*". This resulted in a set of 1,932 captions, ranging from 1 to 22 words in length, with a mean length of 9.0 words.

The queries were obtained by examining a set of randomly chosen magazine images and generating phrases that could reasonably be expected to elicit such images. This resulted in a set of 47 queries, ranging from 1 to 6 words in length, with a mean length of 2.89 words. The relevance judgements were created by first submitting the queries to ANVIL in a variety of high recall configurations to create a pool of candidate results. These were then independently assessed for relevance by two human judges.

Run	**Precision at 10% recall**
Keyword matching (Judge 1)	86%
Keyword matching (Judge 2)	85%
Phrase matching (Judge 1)	95%
Phrase matching (Judge 2)	92%

Table 1: Evaluation results

The query set was submitted to ANVIL and the results evaluated using the standard TREC evaluation procedure. The results for precision at 10% recall are shown in Table 1. ANVIL is aimed at casual users, many of whom may be unwilling to look through a large list of results to find a good one [11]. Therefore, we are mostly concerned with the contents of the first few retrieval results, and precision at 10% recall gives a good indication of this.

7.2 Performance and scalability

The previous test used a database of 1,700 images and 1,932 captions, which is of relatively modest size when compared with many of the typical stock photography collections currently available. Consequently, we need to be concerned with how well ANVIL performs when the database size is scaled up to more realistic proportions. We therefore ran a further set of trials, measuring the time taken for database creation and querying as the number of captions was increased.

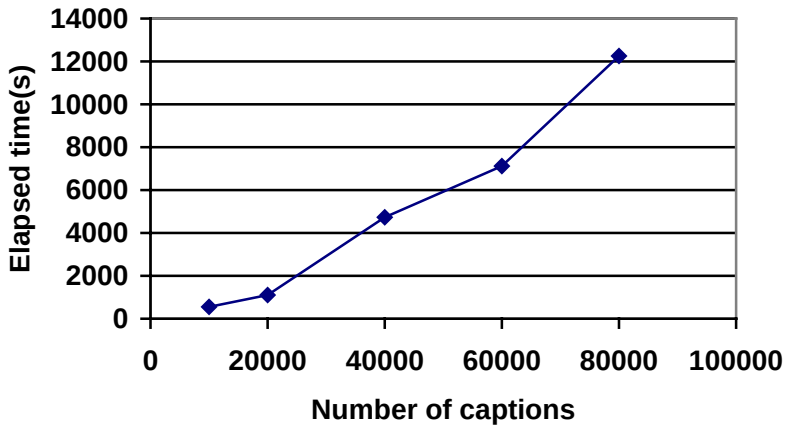

Figure 5: Database creation time against number of captions

As with the previous evaluation, suitable standardised test data is scarce, and again we had to create our own test set. We did this by randomly extracting noun phrases from the parsed Wall Street Journal corpus, with a minimum length of 3 words and a maximum of 60 words. These would act as our caption data. We then grouped them into collections ranging in size from 10,000 to 80,000 captions, and built databases from them. The time taken to create these databases on a Sun Ultrasparc 5 machine running Solaris is shown in Figure 5, as elapsed time. The resultant file sizes are shown in Figure 6. It can be seen that both quantities scale up fairly linearly with the number of captions, which is what we would hope for.

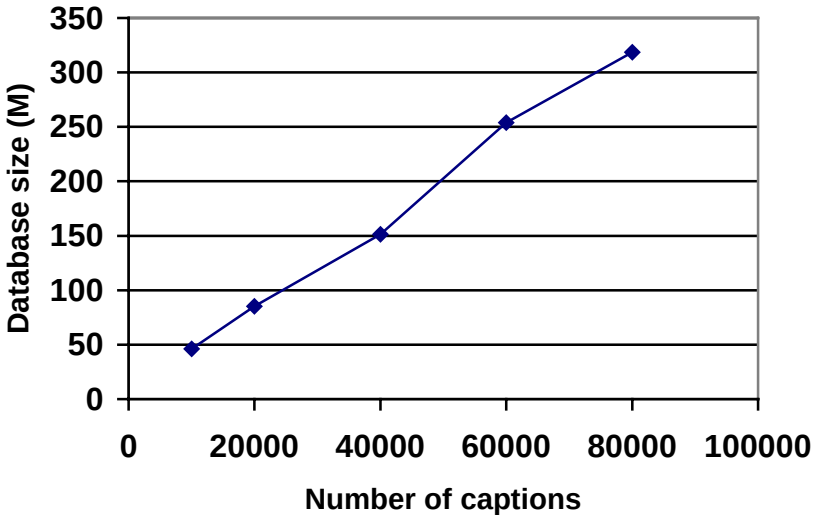

Figure 6: Database size against number of captions

The time taken to execute 50 queries is shown in Figure 7. For this run, ANVIL was configured to use Smeaton & Quigley's algorithm for keyword matching followed by phrase matching. If vector cosine is used for the keyword matching, ANVIL runs significantly more slowly. The graph shows that the correlation is again roughly linear. The overall processing time is limited by controlling various configuration parameters. For example, the number of candidate captions passed through the system can be limited, or the minimum score thresholds can be raised. However, detailed profiling of the run time behaviour of the system shows that the bottleneck is actually in the database access rather than the language processing. Consequently, further performance tuning efforts would be best focussed on improving database caching or using a more compact external representation.

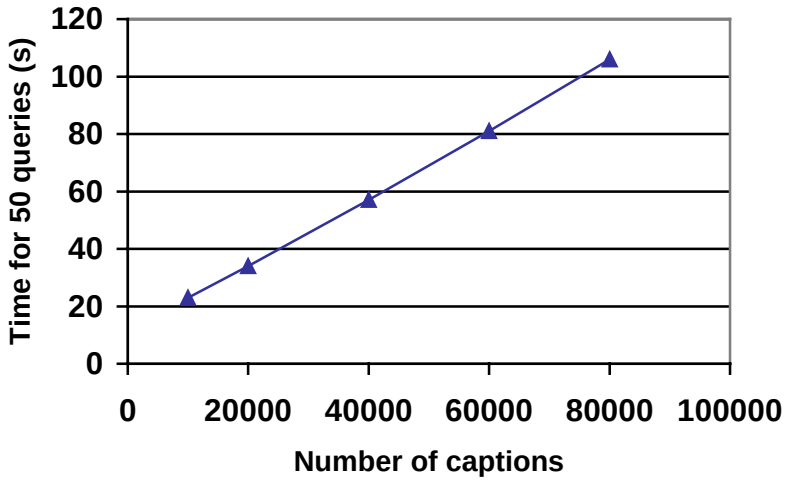

Figure 7: Querying time against number of captions

7.3 User Interface Pilot Study

The two interfaces described in Section 6 were evaluated in a pilot study[1] [12]. Previous studies have shown that the use of browsing increases as the specificity of retrieval tasks decreases [13]. Therefore, systems that provide good support for browsing should facilitate more abstract tasks more effectively than systems that do not. The aim of the study was therefore to test whether the 2D interface provided better support for more abstract tasks than the 1D interface. The levels of abstraction were defined using Eakins' classification scheme [14], which consists of the following three levels:

1. Retrieval tasks based on visual features, i.e. colour, texture, shape, etc.
2. Retrieval tasks based on derived attributes, i.e. objects of a given type ("*a lake at sunset*") or named entities ("*St. Paul's Cathedral*")
3. Retrieval tasks based on abstract attributes, i.e. named events or themes ("*The FA Cup Final*") or emotive responses ("*romantic love*")

No	Search task description
1	Can you find two images that portray the feeling of 'countryside'?
2	Imagine you are writing an article for a magazine that reports on the latest results of the World Chess Championship. You want to accompany the text with an image. Use the system to find two appropriate images.
3	Can you find two images that portray 'friendliness'?
4	Can you find two images that portray 'romantic love'?
5	Imagine that you were given the task to illustrate a cover for a book that deals with alienation. Can you find two alternatives that portray the feeling of 'remoteness in the big city'?
6	Do you consider the technological advances of recent times good or bad for our society? Depending on your belief, can you find two images that support your opinion?
7	Find two images of your favourite fruits.
8	Can you find two images that are good representations of 'ruins'?

Table 2: Retrieval tasks

[1] performed in collaboration with the University of Glasgow

The study took the form of a controlled experiment using a between-groups design, with the user interface as the independent variable: either UI1 (the 1D interface) or UI2 (the 2D interface). 16 participants were recruited from the University of Glasgow, and each underwent the following procedure:

1. **Orientation**: an explanation of the experiment and its purposes
2. **Training**: an opportunity for the participant to briefly explore the UI and ask questions
3. **Tasks**: the participant followed a set of eight written tasks and would attempt to retrieve the two most appropriate images for each.
4. **Questionnaire**: upon completion of each task the participant would fill in a questionnaire scoring the UI on a set of nine 7-point semantic differentials and three 5-point Likert scales.

The experiment was conducted in Glasgow University's usability lab, on a Pentium PC with an Internet connection to the ANVIL server running remotely at Canon Research Centre Europe. Table 2 shows the eight retrieval tasks, each of which was designed to correspond to either level 2 or level 3 in Eakins' classification scheme. The questionnaire was designed to elicit measures of the effectiveness, efficiency and user satisfaction of each interface. The semantic differentials and Likert scales for the questions are summarised in Tables 3 and 4 respectively.

In addition to the questionnaire data, the experiment gathered a limited amount of quantitative data, in the form of the time taken and the number of textual queries required to complete each task. The Mann-Whitney (U) test was used to assess the significance of the findings. Table 5 summarises the results that were statistically significant.

Tested	**Differential 1**	**Differential 2**
Was the task…?	Clear	Unclear
	Simple	Complex
	Interesting	Boring
Was the search process…?	Simple	Complex
	Pleasant	Unpleasant
	Easy	Difficult
	Restful	Tiring
Were the retrieval sets…?	Relevant	Irrelevant
	Useful/Appropriate	Useless/Inappropriate

Table 3: Semantic differentials

Statements
1. I had a clear idea of the image that would satisfy the task's requirement (a well-defined information need)
2. I am positive that I saw all the alternatives relating to the task given (level of recall)
3. I am satisfied with the image(s) I found to help me with the task at hand (satisfied with overall outcome of the search process for particular task)

Table 4: Likert scale statements

Finding	Level of significance
Participants using UI2 tended, on average, to regard the tasks in a more positive way than those using UI1 (means: UI2=2.4, UI1=3.4)	0.005
Participants using UI2 characterised the search process as easier than those using UI1 (means: UI2=2.7, UI1=3.0)	0.05
Participants using UI1 were more certain that they saw all the possible images relevant to the task than those using UI2 (means: UI2=3.0, UI1=2.2)	0.025
Participants using UI2 used a smaller number of textual queries per task than those using UI1 (means: UI2=3.7, UI1=4.0 queries)	0.025
Participants using UI2 took, on average, less time to complete each task than those using UI1 (means: UI2=4.1, UI1=5.2 minutes)	0.05

Table 5: Statistically significant findings

The questionnaire was designed to elicit data on the effectiveness, efficiency and user satisfaction of the interface. The results for each of these criteria may be summarised as follows:

- **Effectiveness**: this criterion considers the accuracy and completeness with which users achieve specified goals. The participants rated UI1 significantly better than UI2 in presenting all the relevant alternatives in relation to a given task ($p=0.025$)
- **Efficiency**: this criterion considers the resources expended in achieving the specified goals. Participants needed significantly less time to complete each task when using UI2 ($p=0.05$)
- **User satisfaction**: this criterion considers the comfort and acceptability of the system to its users. Participants indicated that UI1 was simpler, easier and more restful. However, UI2 was assessed as providing a significantly more pleasant experience ($p=0.05$).

In order to reject the null hypothesis, we would expect to UI2 to outscore UI1 in all three criteria. Since many of the results proved to be non-significant ($p>0.05$), the null hypothesis cannot be rejected. However, this study was investigative in nature, and the findings at this stage should be considered as exploratory rather than conclusive. We therefore hope to further refine both interfaces and explore the principles underlying each design with more detailed experiments and studies.

8. Conclusions

This paper has described ANVIL, a natural language based system for the retrieval of captioned images. Evidently, ANVIL is different to many other IR systems in that it works with short captions rather than complete documents. This allows us to apply a more detailed NL analysis than may otherwise be the case. In addition, we have not needed to consider the type of discourse-level phenomena that affect the sentences typically found in larger texts. Conversely, larger texts provide more data concerning the topic of a document, which can make a significant contribution to retrieval performance.

Whatever type of approach is used, the process of designing product-level IR systems relies on finding the right balance between the need for accuracy and the computational overheads that this incurs. We also need to balance the needs of different types of user: phrase matching is designed to assist those who can precisely formulate their information need; the 2D interface is designed to assist those who cannot. In ANVIL we hope we have found a good balance, and look forward to further evaluation of its performance as a publicly accessible, live system in the near future.

References

1. Sparck-Jones, K. What is the role of NLP in text retrieval? In Strzalkowski T (Ed.) Natural Language Information Retrieval. Kluwer, 1999.

2. Flank, S. A layered approach to NLP-based information retrieval. Proceedings of the 36th ACL and 17th COLING, 1998, pp 397-403.

3. Fellbaum, C. (ed) WordNet: An electronic lexical database. MIT Press, 1998.

4. Jiang, J, Conrath, D. Semantic similarity based on corpus statistics and lexical taxonomy. Proceedings of ROCLING X, 1997.

5. Smeaton, A, Quigley, I. Experiments on using semantic distances between words in image caption retrieval. Proceedings of SIGIR, 1996, pp. 174-180.

6. Robertson, S. et al. Okapi at TREC-3. Proceedings of the 3rd Text Retrieval Conference, 1994, pp 109-126.

7. Zizi, M. Interactive Dynamic Maps for Visualisation & Retrieval from Hypertext Systems. In Agosti, M, Smeaton A (eds) Information Retrieval and Hypertext. Kluwer Academic, 1996.

8. Beale, R. Foreign Interactions. Interfaces 37 1998; pp. 23-26.

9. Rubner, Y. et al. A metric for distributions with applications to image databases. In Proceedings of the IEEE Conference on Computer Vision, 1998, pp. 59-66.

10. Rodden, K, Basalaj, W, Sinclair, D, and Wood, K. Evaluating a visualisation of image similarity as a tool for image browsing. Proceedings of the IEEE Symposium on Information Visualisation, San Francisco, 1999.

11. Pollock, A, Hockley, A. What's wrong with Internet searching. D-Lib Magazine, March 1997.

12. Tsonis, P. "Image retrieval user study". MSc Thesis, University of Glasgow, 1999.

13. Gupta, A, Santini S, and Jain R, In search of information in visual media. Communications of the ACM, vol. 40, No. 12, 1997, pp. 71-79.

14. Eakins, J. Automatic image content retrieval – how usable is the technology? Int. Journal of Electronic Library Research 1997; Vol. 7, No. 1, pp. 63-88.