MemeFinder: Search engine for unstructured personal meme collections

游子緒
D09922024
d09922024@ntu.edu.tw
- Visual similarity
- User interface
- Report writing

劉穎立
B06901004
b06901004@ntu.edu.tw
- Textual similarity
- Optical character recognition
- Demo video

黃世丞
R09942093
r09942093@ntu.edu.tw
- Text Recognition
- Word2vec
- Server deployment

魏晧國
R09922118
r09922118@ntu.edu.tw
- Data collection
- Pseudo labeling

1 INTRODUCTION

Memes are pictures depicting emotional situations, often with captions. They can be composed and modified by anyone, and they propagate fast on social media. Many fan pages collect, translate, and publish memes, so it's not difficult to find a vast number of memes. However, finding a specific meme remains a challenging task.

Online meme collections on fan pages are usually not equipped with responsive search engines. Beside fan pages, there are also specialized websites for sharing and finding memes, e.g., memes.tw[1] and imgflip.com[2]. The search engines of these websites are usually based on tags from volunteers, which only covers a small portion of the collection. Some users may have personal collections of memes, but tagging and classifying memes takes much human effort.

To solve this problem, we created a search engine, MemeFinder, as shown in Fig. 1. MemeFinder is designed for unstructured personal meme collections, with optional sparse tags. MemeFinder is based on the visual features, the captions, and the optional tags of memes in the collection. We implemented a web-based user interface (UI) and conducted an user study, where 15 users are involved. 45% of the users prefer MemeFinder to Google Images when used to search for a specific meme.

Since tags usually have a small coverage in a collection, pseudo labeling is used to predict the tags for unlabeled memes. In the user study, we also evaluated the effectiveness of including tags in the retrieval model. 57% of the users found the target meme with the OCR+tag retrieval model faster than the OCR-only retrieval model.

[1] https://memes.tw/
[2] https://imgflip.com/memetemplates

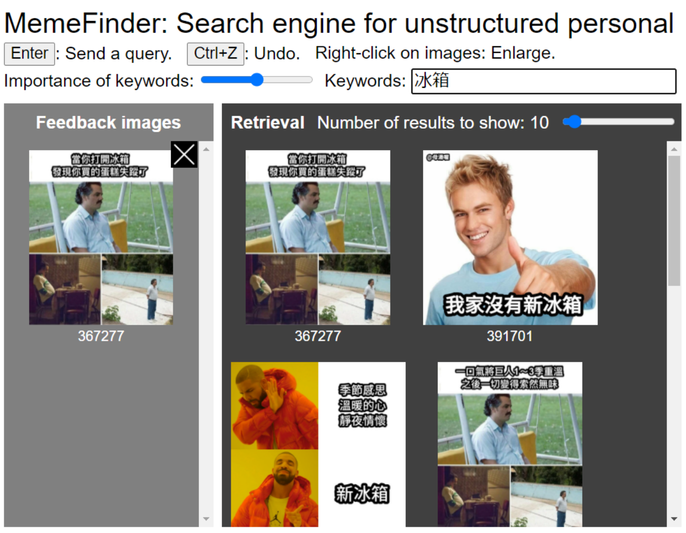

MemeFinder: Search engine for unstructured personal
Enter: Send a query. Ctrl+Z: Undo. Right-click on images: Enlarge.
Importance of keywords: Keywords: 冰箱

Figure 1: The MemeFinder UI with an example query. The user input consists of a list of feedback memes, a list of keywords, and a slider controlling the importance of textual features.

2 RELATED WORK

We did not find existing works about meme retrieval, but a work on retrieval with visual and textual features is found [1].

The purpose of [1] is to find items in an online shopping platform, of the same brand and same type as the query image. The same merchandise is often presented differently by different sellers, and this makes comparing prices difficult. In [1], a 2-stage retrieval scheme is proposed. The query is a single image. In the first stage, the visual features extracted with an attention-based CNN model are compared, and a ranked list is produced. The text description of the first few items is taken as relevance feedback to find candidates for the second stage. In the second stage, candidate items are again ranked by visual features.

Memes differ from merchandise descriptions in many ways. First, memes based on a common template usually have very similar visual features, while photos of the same merchandise taken from different viewpoints may look totally different. Second, the captions in a meme is much shorter than a merchandise description, and may not be related to the visual features at all. Due to these observations, and due to the different purposes, our retrieval model is largely different from the one in [1].

3 METHODOLOGY

Since memes are images, a straightforward feature to extract is the visual feature, which can be easily done by pre-trained image classification models.

The captions in a meme are recognized by a NN model, and the cropped text region are fed into an optical character recognition (OCR) engine.

The tags are only labeled for a small portion of the collection, so pseudo labeling is used to predict the unknown tags. Our strategy is to find neighboring memes for each unlabeled meme based on visual or textual similarities, count the tags in the neighborhood, and assign the tags with counts over a threshold.

The captions and tags are combined to calculate the textual similarities with the query text. The final scoring function is a weighted sum of the textual similarity and the visual similarity.

The details of each component will be described in the following subsections.

3.1 Visual similarity

We use the pre-trained EfficientNet-B0 [2] model to extract the visual features of the meme images. The feature vector is obtained from the output of the final linear layer. Note that the feature vectors are usually taken from the output of the last convolution layer, but we did not adopt this convention. The reason is that, at an early stage of this project, we planned to use the image class labels in the retrieval system, but we did not have time to explore this option.

We adopt cosine similarity as the distance measure. The reason for choosing cosine similarity is from an observation on the feature vectors. The L2 norm of the vectors are distributed around a mean value and the deviation is small relative to the mean, as shown in Fig, 2. In simple words, the vectors are distributed like the shell of a hypersphere, which makes cosine similarity is a suitable choice.

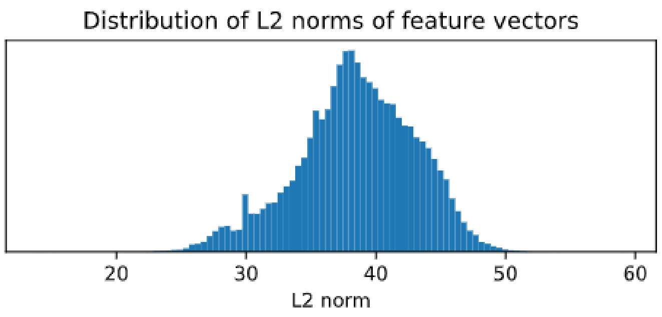


Figure 2: The distribution of lengths of visual feature vectors.

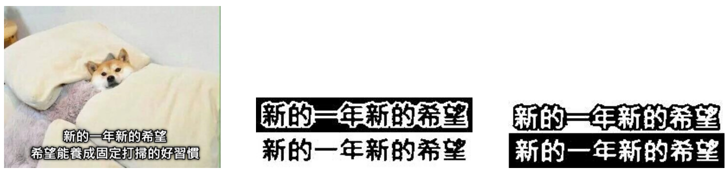
3.2 Text detection

We use pre-trained CRAFT [3] model to crop text images from the original image. CRAFT is a deep neural network optimized for detecting text bounding boxes in images. Though it was trained on an English corpus, we have empirically verified that it works charmingly well on our Chinese meme dataset.

3.3 Optical character recognition

Tesseract OCR [4] is a powerful OCR engine developed by Google. It can achieve high accuracy with fast inference speed, which is crucial in our retrieval system. However, we have discovered that the accuracy highly relies on the quality of input images. When the words are in black text with white background, so called the *positive text*, the accuracy will be much higher than the inverse ones (white text with black background), which are referred to as *negative text*. Fig. 3. shows the difference between these two situations. Notice that the word ”background” we mention here is the part directly surrounding the text. So we cannot determine the background color by simply choosing the color with largest coverage.

To enhance the performance of the OCR, we will first binarize the cropped image, then apply Tesseract OCR on both the binarized cropped image and the inverted one, and choose the one with higher confidence score.



(a) Original image (b) Positive text (c) Negative text

Figure 3: Positive text and negative text

3.4 Pseudo labeling

Before pseudo labeling, we need to weed out those tags that appeared many times but were not helpful in retrieval. We count the frequency of tags, and consider those high-frequency tags such as ” 梗圖” ” 迷因” as stop words.

To generate pseudo labels, for every meme we aggregate the tags of their 100-nearest neighboring memes, in terms of visual similarity. We count the occurrence of tags, and label the center meme with those tags that occurs over 8 times.

It seems that some tags have similar meanings while they are considered different tags. For example, the meaning of " 台大" and " 臺大" are exactly the same, but they are still considered different tags. Moreover, some memes have similar texts and should be labeled with similar tags. To overcome this problem, we tried extending our pseudo labeling technique by generating a sentence embedding for each meme. However, the t-SNE result of the embeddings showed that the textual features were not properly extracted, so only the visual features are utilized eventually.

3.5 Textual similarity

For each document, we first apply word segmentation with CkipTagger[3] [5]. CkipTagger is an open-source NLP tool which can do word segmentation and serveral NLP tasks. CkipTagger not only achieves better performance than the well-known word segmentation system jieba[4], but also shows great ability on segmenting Taiwanese style sentences.

Then we use tf-idf to calculate the weighting score of each term in the documents. To be precise, consider a document

$$D_i = \{w_{i1}, w_{i2}, ..., w_{it}\} \in D$$

, where w_{ik} is the weight of the term T_k in D_i and t is the number of unique terms in D_i. We calculate the term frequency score $tf(f_{ik})$ with the following formula:

$$tf(f_{ik}) = \frac{1 + log_2(f_{ik})}{1 + log_2(avg(f_{ik}))}$$

, where f_{ik} is the occurrence frequency of term T_k in document d_i. The inverted document frequncy score is calculate by the following formula:

$$idf(N, n_k) = log_2(\frac{N+1}{n_k})$$

, where N is the total number of documents in the corpus, and n_k is the nubmer of documents with term T_k present. The term weight is finally calculated by:

$$w_{ik} = tf(f_{ik}) \times idf(N, n_k)$$

. For each query, we also calculate the weighting score of each term. Next, the documents are ranked by the cosine similarity between the tf-idf vectors of the query and documents.

As for pseudo labeling, we simply concatenate the original texts with the tags and apply the same procedures above.

3.6 Retrieval model

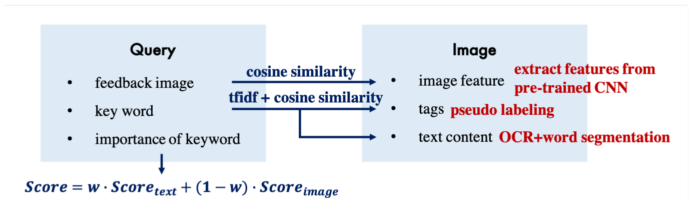


Figure 4: The system flow of MemeFinder.

Fig. 4 shows the system flow of Memefinder. A query consists of a list of feedback images, a list of keywords, and a weight $w \in [0, 1]$ controlling the importance of textual similarity. The final ranking score is the weighted sum of the visual similarity $s_v \in [0, 1]$ and the textual similarity $s_t \in [0, 1]$:

$$s = w \times s_t + (1 - w) \times s_v$$

.

At the home page of MemeFinder is a set of 20 memes. These memes serve as initial sources of feedback images. The set is found by clustering the visual features of the collection by k-means, where $k = 20$. A meme around the center of each cluster is added to the set.

When the user wants to find a meme in his memory, it can start from searching with a text query and an empty list of feedback images. The retrieval results may contain images similar to the target meme, and those similar images can be added to the list of feedback images to refine the query. Alternatively, the user can also start with similar images chosen from the homepage and an empty text query, and refine the query iteratively. In either case, the user plays an important role in the feedback loop.

4 EXPERIMENTS

4.1 Dataset

The dataset for evaluation is collected from memes.tw[5]. A total of 132272 memes is in the collection, where 55% of the memes have tags. The collection contains groups of images with the same templates. Some popular templates form large groups, which justifies the need of a search engine. Table 1 shows the most popular templates.

[3] https://github.com/ckiplab/ckiptagger
[4] https://github.com/fxsjy/jieba
[5] https://memes.tw/

Table 1: The most popular templates.

Template	Amount
美國隊長電梯	4705
Drake No Yes	4601
鸚鵡兄弟	3036
我就爛	2747
黑人笑	2320
亞洲人之恥	2012
蝙蝠俠	1830
American Chopper Argument	1817

4.2 Procedure

We give each subject 20 randomly sampled target memes and ask them to find the memes with the implemented search engine. The target memes are split into 2 sets of 10 memes. The first set is evaluated with the OCR+tag retrieval model, and the second set is evaluated with the OCR-only retrieval model.

In a real-world use case, users are not expected to remember the exact details in the target meme, so we tried to reproduce this state. We mask a random quadrant or the target meme, and show the masked meme to the subject for 1 second before retrieval starts. The subject is allowed to skip a meme if it think it is too difficult to find.

To evaluate the implemented search engine, we measured the following indicators for finding a specific meme in each set:

- The spent time.
- The number of queries.
- The number of skips performed.
- The perceived spent time.
- The perceived number of queries.
- The perceived difficulty.

4.3 Result

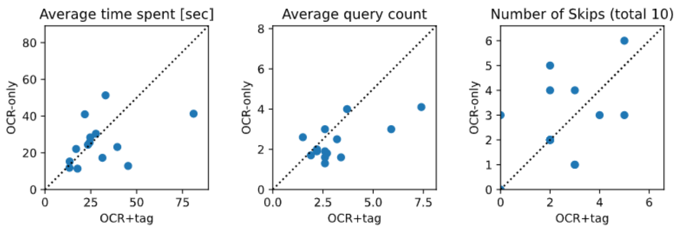


Figure 5: Objective indicators of 15 subjects.

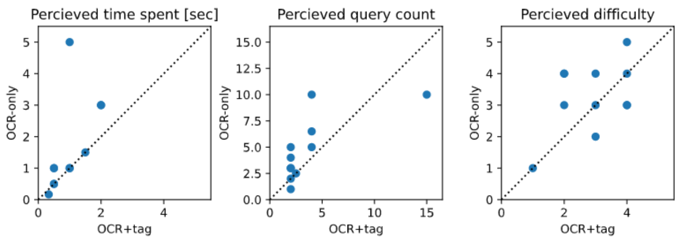


Figure 6: Subjective indicators of 11 subjects.

15 Subjects are involved in the user study, where 11 subjects submitted the questionnaire. Fig. 5 shows the objective indicators, and Fig. 6 shows the subjective indicators collected in the questionnaire.

In Fig. 5, 57% of the subjects spent less time with the OCR+tag version. However, it can be observed that some users spent much more time and more queries when searching with the OCR+tag version, which is contrary to our expectation. The reason may be due the order of the tests. The first set is evaluated with the OCR+tag version, and the subjects may spent more time to get familiar with the search engine. This hypothesis is supported by the subjective indicators in Fig. 6. The users feels they spent less time and queries with the OCR+tag version. Out of the 11 subjects, 5 of them feels that tests with the OCR+tag version is simpler, while 3 of them feels more difficult.

In the questionnaire, we asked the subjects which search engine they prefers when searching for memes. With the OCR+tag version, 45% of the users prefers MemeFinder.

5 CONCLUSIONS

We created MemeFinder, a search engine designed for unstructured meme collections. Since typically only a small proportion of these collections have appropriate tags, pseudo labeling is used to predict the missing tags. A user study validated the effectiveness of adding the pseudo-labeled tags in the retrieval model.

REFERENCES

[1] Jingwen Hou, Sijie Ji, and Annan Wang. Attention-driven unsupervised image retrieval for beauty products with visual and textual clues. 10 2020.

[2] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks, 05 2019.

[3] Youngmin Baek, Bado Lee, Dongyoon Han, Sangdoo Yun, and Hwalsuk Lee. Character region awareness for text detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9365–9374, 2019.

[4] Ray Smith. An overview of the tesseract ocr engine. In *Ninth international conference on document analysis and recognition (ICDAR 2007)*, volume 2, pages 629–633. IEEE, 2007.

[5] Peng-Hsuan Li, Tsu-Jui Fu, and Wei-Yun Ma. Why attention? analyze bilstm deficiency and its remedies in the case of ner. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8236–8244, 2020.