An Image-News Matching System

Kexin Zhu and **Zhilin Han** and **Zhengneng Chen**
Rensselaer Polytechnic Institute
Troy, NY 12180, USA
{zhuk4, hanz3, chenz11}@rpi.edu

Abstract

Inspired by the lacking of up-to-date image-description dataset, and to deepen our understanding on state-of-art natural language processing techniques and their mathematical principles, we developed a system that can be used to matching news and news photos. It includes image captioning, keyphrases extraction, events extraction. The system will take a set of images and a set of news as input, then generate cations for images, extract keyphrases and events from news to match news with images. Our goal is to automatically generate images set with **detailed and up-to-date** description to enrich dataset for both natural language processing and computer vision. The main idea of image captioning system is from Vinyals et al. (2015). We used PyTorch to re-implement part of the image captioning model from that paper.
Keyphrase extraction plays an important role in the task of indexing, summarization, clustering, categorization. The ideas is inspired by El-Beltagy and Rafea (2009) The event extraction model was built with the idea of Joint Event Extraction via RNN (Nguyen et al., 2016).

1 Introduction

The state-of-art models of image captioning and event extraction achieve their excellent performances by introducing Deep Learning (DL) into the statistical learning procedure. Noting that the latest version of a capstone textbook in NLP area, Foundation of Statistical Natural Language (Manning and Schütze, 1999), added lots of Deep Learning models into it's content to follow up the latest researches. For the tasks of Event Extraction and Imae Caption, a variety of conference papers show the potential of DL and examine performances between neural networks and traditional statistical methods.

As we learn NLP, Prof. Ji showed basic ideas of different tasks in NLP area as well as how capable DL is. We were instructed to handle assignments' requirements using some techniques from Computational Linguistics like word embedding, part-of-speech (POS) tagging and syntactic bracketing as well as apply DL models built in Theano and PyTorch to process text data with POS taggers and in word embedding vector space.

Known that DL plays an important role in NLP along with our interests in Image Caption and Event Extraction, we decided to implement an Image Caption model using the idea of Vinyals et al. (2015) and an Event Extraction model using the idea of Li and Ji (2014a) to help us understand how meaningful models work and be familiar with challenges that we are going to meet in NLP researches.

2 Usage

2.1 Package installation

numpy, h5py, scipy, nltk, networks,sklearn, unidecode, PyTorch, lxml
Please install all above package in python2.7 environment

2.2 Command Line

Since there's pretrained CNN model required for our image captioning part, the size of our project exceeds 400 MB. If you would like to try our system on your own (new images and news), you will have to download it from https://drive.google.com/open?id=1lYuDlpTxOIKPUvGmAVl2ZgYDiVqH7P2d.

2.2.1 Evaluate Provided Data

```
python final_eval.py
```
Provided test data located in "./test_data"

2.2.2 Evaluate New Dataset

```
python final_eval.py
--image_folder <YOUR DATASET DIR>
--new_dataset 1
```

Make sure to replace <YOUR DATASET DIR> with folder that contains your new images(jpg format) and news (text format). The folder must be under root directory **"KexinZhu_ZhengnengChen_ZhilinHan_ImageNewsMa**

3 Tasks

3.1 Image Caption Task

The first problem we solved is generating descriptions from an input image. We used encoder-decoder model to finish this task, where the encoder is a Convolutional Neural Network and the decoder is a Long Short Term Memory unit. CNN can extract features from images, while LSTM can generate a sequence of words to form a sentence.

3.2 Event Extraction Task

Since our Event Extraction model will only focus on event trigger classification, we use the results of arguments candidates generated from RPI Joint Information Extraction System (Li et al., 2013, 2014; Li and Ji, 2014b). Then there are two main challenges remained: Preprocessing data into trainable format and training a bi-directional LSTM model to predict event triggers and their associating arguments.

3.3 Keyphrase Extraction Task

For one piece of news, extract the N best selection of keyphrases to describe the news. In order to implement the model, we modified and improved from the open source python-based keyphrase extraction toolkit(Boudin, 2016), also combine the method keyphrase extraction algorithm (El-Beltagy and Rafea, 2009) to improve the performance. Here we choose twenty keywords to describe the news for further matching system.

4 Model

4.1 Image Captioning Model

The image captioning part is a neural and probabilistic framework to generate descriptions from images. We used encoder-decoder model to design it. The encoder is a Convolutional Neural Network, which can produce a rich representation of the input image by embedding it to a fixed-length vector (Sermanet et al., 2013), and the decoder is a Long Short Term sentence generator.

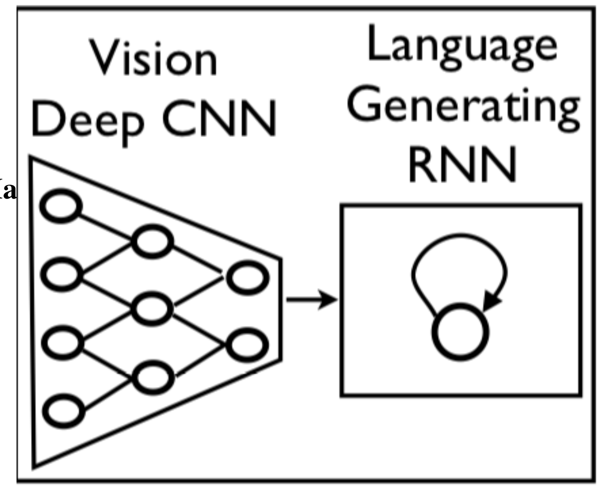


Figure 1: Our image caption model generates complete sentences in natural language from an input image

It's natural to propose to directly maximize the probability of the correct description given the image by using the following formulations (Vinyals et al., 2015)

$$\theta* = argmax \sum_{(I,S)} \log P(S|I;\theta)$$

where I represents input images, S is one of its correct descriptions, and θ is the weights of our model. If we use $S_0, S_1, S_2, ..., S_n$ represent every words in sentence S, we can get a new formulation

$$\log P(S|I) = \sum_{t=0}^{n} P(S_t|I, S_0, ..., S_{t-1})$$

If we model $P(S_t|I, S_0, ..., S_{t-1})$ with a Recurrent Neural Network, where the variable number of words we condition upon up to t-1 is expressed by a fixed length hidden state or memory h_t. This memory is updated after seeing a new input xt by using a non-linear function f

$$h_{t+1} = f(h_t, x_t)$$

where f is actually a Long Short Term Memory unit.

4.1.1 Convolutional Neural Network

Since this is a project report for natural language processing course. I'm not going to show too much details of extracting features from images. We use 2 convolution layers and 3 fully connected layers to downsample every image into a fixed dimension vector. Then send the feature vector into LSTM unit.

4.1.2 LSTM Unit

The memory cell of LSTM will encode every inputs in each time step. Based on the paper of Vinyals et al. (2015), we can make the memory cell achieve this behavior by adding "gates" - layers which are applied multiplicatively. By adding the gates onto the unit, the memory cell will be able to choose whether keep a value from a gated layer or not (depends on the gate is 1 or 0).

In our implementation, after applying sigmoid function, in each step the unit will produce a probability distribution over the current word. To implement this LSTM unit, I study lots of example from tutorial of PyTorch and the source code of Vinyals et al. (2015), which was implemented with TensorFlow.

4.1.3 Training

We used MS-COCO dataset to train this image captioning model.

The LSTM unit was trained to predict a sequence of words as defined by $P(S_t|I, S_0, ..., S_{t-1})$. To implement this process, it's easier to think in an unrolled model,

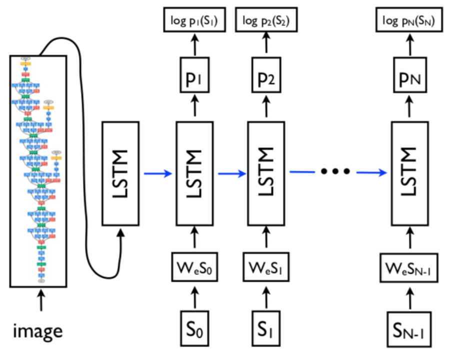


Figure 2: Down-sampling image into feature vector, and LSTM unit generating sequence of words. All LSTMs are actually the same unit, sharing same parameters.

If we use I to represent input image and $S = (S_0, S_1, ..., S_N)$ to represent one of the sentences that could briefly describe the image, we can specifically unroll the procedure as

$$\begin{aligned} x_{-1} &= CNN(I) \\ x_t &= W_e S_t, t \in \{0...N-1\} \\ P_{t+1} &= LSTM(x_t), t \in \{0...N-1\} \end{aligned}$$

4.1.4 Loss Function

We use the sum of the negative log likelihood of correct word at each step as loss function. As following,

$$Loss(I, S) = -\sum_{t=1}^{N} \log P_t(S_t)$$

4.2 Event Extraction model

4.2.1 Data preprocessing

Preprocessing data needs patience and an insight from higher level since Nguyen et al. (2016) used some trick on sentence encoding and representation in their paper. Here I simplify the representation into pure word embedding vectors. Embedding vector of entity mentions is useful for arguments prediction (Li et al., 2013) and hence are deprecated in order to simplify the difficulty of implementation.

Then for a sentence $\mathbf{W}$, we get the word embedding vector $\mathbf{x}_i$ for each token $\mathbf{w}_i$. Word embedding vectors are achieved by looking up a pre-trained word embedding table which is trained from English Wikipedia corpus. This table is the same as the one we used in Assignment 2.

4.2.2 Bi-directional LSTM

Given a input vector, the representation of a sentece, $\mathbf{X} = (x_1, x_2, \cdots, x_n)$. We are going to compute hidden vector a_i recurrently by using a non-linear transformation function $a_i = \Phi(x_i, a_{i-1})$. Then the computation is performed on each entry of $\mathbf{X}$ and we have one of hidden vector sequence which is $\overrightarrow{RNN}(\mathbf{X}) = (a_1, a_2, \cdots, a_n)$. Here we are using the characteristic of RNN that each hidden vector could catch the context information from x_1 to x_i. Noting that this hidden vector can only accumulate those context information from the start of sentence. Thus there are one more hidden vector sequence needed.

Another recurrent computation is performed to compute the reversed hidden vector

$a'_1 = \Phi(x_i, a'_{i+1})$. Then we have the other RNN $\overleftarrow{RNN}(\mathbf{X}) = (a'_1, a'_2, \cdots, a'_n)$ where each hidden vector a'_i contains the context information from n to i.

For the non-linear transformation function Φ, we use long-short term memory units (LSTM) to prevent the problem of "gradient vanishing" (Bengio et al., 1994).

4.2.3 Model training

Noting that representation vector X is word embedding vectors of sentence. Let $\mathbf{T} = t_1, t_2, \cdots, t_n$ denotes trigger subtypes. Model's training process is to minimize the negative log-likelihood function C for triggers:

$$\begin{aligned} C(T, \mathbf{X}) &= -\log P(T|X) \\ &= -\sum_{i=1}^{n} \log P_{i;t_i}^{triggers} \end{aligned}$$

The loss function is default one which is cross entropy. To speed up the training process, I use stochastic gradient descent (SGD) as a optimization. Our event extraction model is trained on ACE-2005 dataset. After training is completed, the performance of model is Precision-71.9, Recall-61.5 and f-score-66.3

4.2.4 Trigger classification

For the sentence or text we would like to predict triggers and classify an event type and subtype for triggers, we first compute the feature representation vector $mathbfR$ of sentence. This feature representation is consisted of 3 parts:

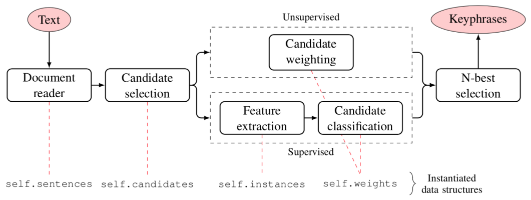
- hidden vector $\mathbf{H} = (h_1, h_2, \cdots, h_n)$
  Each entry of hidden vector is computed using $\overrightarrow{RNN}$ and $\overleftarrow{RNN}$ as we did previously. This hidden vector is used to restore those context information suggested in sentence.

- local context vector $\mathbf{L}$
  This vector is a bit confused at first. However, we are covered with the topic of n-grams in lectures. Along with the material provided in Nguyen et al., I view it as a local embedding vector by looking up nearby context with fixed windows length.

- memory vector $\mathbf{G}$ from previous step

Then we concatenate three vectors to get the representation desired and then put it into a feed-forward neural network F. Output layer of this FFNN is a softmax layer. Then we compute the probability distribution of event types and subtypes which are determined by maximum likelihood.

4.3 Keyphrase Extraction Model



4.3.1 Training Data

ACM Digital Library(Conference and Workshop papers)

4.3.2 Candidate Identification

Use a brute-force method to consider all the words as candidates. Then filter the candidates into a candidate set. Remove all the stop words and punctuations by NLTK package. Discard all the numeric words, and word with too many characters or too less characters. Standardize all the words from changing them to the uniform form, which is to transform each adjective, noun, adverb and past tense to its rootBoudin (2016). So that all the commonly used words are removed and the form for each word is the same.

4.3.3 Keyphrases Selection

To select the best keyphrases for news, we choose the highest score of all the candidates. Score all the candidates and choose the N-highest confidence words. First use the frequency statistics, to get the relative frequency of each candidate. Here we use TF-IDF, short for term frequency inverse document frequency, to score the candidates. The basic idea is to count the frequency of the candidate in the document, and offset by the frequency of the word in the corpus to adjust the candidates frequency. For the term frequency, here we use the raw count of the occurrence number divides to the total word number in the news.

$$tf(t, f) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}$$

$f_{t,d}$: raw count of the candidate t in document d

For the Inverse document frequency, use the formula

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

N: total number of documents in the corpus N = $\{|D|\}$
$|\{d \in D : t \in d\}|$: number of documents where the term t appears

Then score for each candidate is $td \times idf \times \alpha$. F-score for TFIDF method is 16.4, precision is 20.0 and recall is 14.1 El-Beltagy and Rafea (2009). However, some researchers point out that the high frequency candidates are not necessary the best keyphrases to describe the news. So then use a unsupervised machine learning method to rank the keyphrases. We use the topic ranking to score the candidates and choose the keyphrases. First choose the topics by vectorizing candidates, to transform each candidates to a vectorized representation of candidates. Then calculate the number of clusters and form flat clusters to choose the topics. After choosing the topics, build a topic-based graph by connecting all the nodes, loop over all the topics, the weight for the edges is $\frac{1}{|node_i - node_j|}$, then normalize and adjust the weight for the topic-based graph. Keyphrases are produced by extracting the first occurring candidate of the highest ranked topics. F-score for topicrank is 12.6, precision is 15.6 and recall is 10.8. Since this two methods are independent and focus on different features of the words, we combine the two methods to get the keyphrase. After implementing two methods and get normalized weight for each candidates, add the weight of TF-IDF for the first sentence in the document, since it is the title of the news. Then join these two weights together by multiply them, and select the N-highest weight candidates as the keyphrases.

4.4 Matching Results

Use raw counts to match image captions and keyphrase, event extraction. Combine all the triggers, events and keyphrases together as news short description, for each phrase in the lists, if it is in the image caption, then count increment. Choose the highest counts news to match the image.

5 Conclusion

5.1 Work Distribution

5.1.1 Zhilin Han

a) Implements image captioning part.
b) Implements parser program to parse all output text file to JSON file.
c) Designs prototype matching system (using simple word counting, achieve about 25% correctness).
d) Implements final evaluation program.

5.1.2 Kexin Zhu

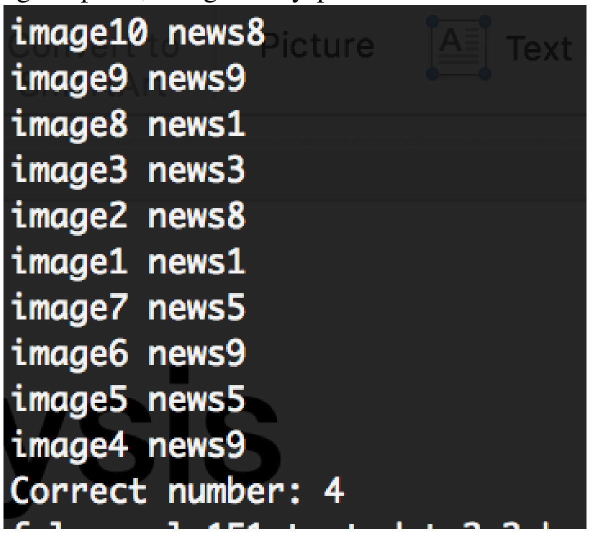
a) Implements keyphrases extraction part
b) Applies word embedding to improve matching result by 50%
c) Finds test data on news websites and tests the result

5.1.3 Zhengneng Chen

a) Implements event extraction part
b) Requests event arguments from RPI Joint IE system
c) Writes up abstract and introduction

5.2 Result

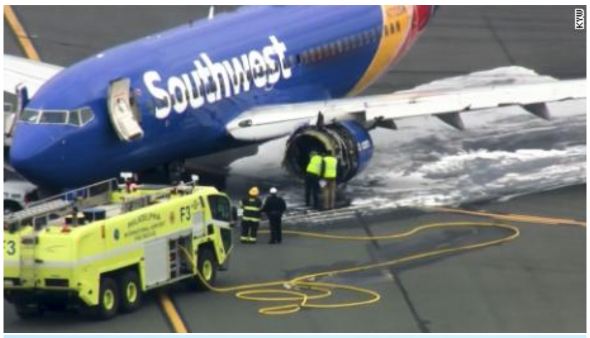
For concise, we choose ten sets of news and images. After applying word-embedding to find similarity between triggers, keyphrases and image caption, we get forty percent of correctness.

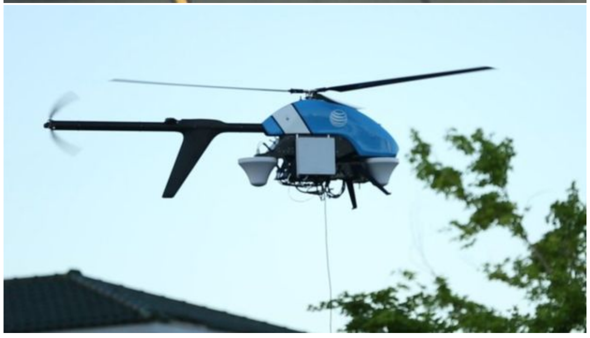


```
image10 news8
image9 news9
image8 news1
image3 news3
image2 news8
image1 news1
image7 news5
image6 news9
image5 news5
image4 news9
Correct number: 4
```

5.3 Error Analysis

We think it is a satisfying result, since some mistakes also make sense.





There is one piece of news is about southwest airplane emergency landing, it should match with the first image, however, in our result, it matches with the second image. From the image caption, the result for first image is ”a large air plane on a run way”, while the result for the second image is ”a plane is flying in the air with a sky background”. Both captions are possible to be matched with the piece of news. So sometimes, we can only match them manually.

5.4 Further Improvement

It is hard to improve the results since one simple news picture is not accurate enough to reflect the news. The best idea to improve the matching correctness is to put more images to match one piece of news. At the same time, through our project, we hope to enrich the datasets for both computer vision part and NLP part, if the CV datasets are abundant enough, the image caption parts will also be improved in order to get a better result.

Acknowledgement

First, we would like to appreciate the instruction of this semester and plenty of precious advices and suggestions on this project from Professor Ji. Also, thanks to our TA, Diya, for grading our assignments and this final project. Additionally, thanks to Boliang and Spencer from RPI Blender lab for providing us with tutoring on PyTorch.

References

Y. Bengio, P. Simard, and P. Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166.

Florian Boudin. 2016. pke: an open source python-based keyphrase extraction toolkit. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations*, pages 69–73, Osaka, Japan. The COLING 2016 Organizing Committee.

Samhaa R. El-Beltagy and Ahmed Rafea. 2009. Kp-miner: A keyphrase extraction system for english and arabic documents. *Information Systems*, 34(1):132 – 144.

Qi Li and Heng Ji. 2014a. Incremental joint extraction of entity mentions and relations. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 402–412. Association for Computational Linguistics.

Qi Li and Heng Ji. 2014b. Incremental joint extraction of entity mentions and relations. In *ACL (1)*, pages 402–412. The Association for Computer Linguistics.

Qi Li, Heng Ji, Yu Hong, and Sujian Li. 2014. Constructing information networks using one single model. In *EMNLP*, pages 1846–1851. ACL.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *ACL (1)*, pages 73–82. The Association for Computer Linguistics.

Christopher D. Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, Massachusetts.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016. Joint event extraction via recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Pierre Sermanet, David Eigen, Xiang Zhang, Michael Mathieu, Rob Fergus, and Yann Lecun. 2013. Overfeat: Integrated recognition, localization and detection using convolutional networks.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3156–3164.