IDENTIFICATION OF EVENT IN HUMAN OBJECT

ARINDAM KUNDU CSE/2013/001

PROLAY KUMAR SAHA CSE/2013/002

SAIKAT MONDAL CSE/2013/003

SRIJIT ROY CHOWDHURY CSE/2014/L-11

UNDER THE GUIDANCE OF

DR. ANUP KR. KOLYA

PROJECT REPORT SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF

BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING

RCC INSTITUTE OF INFORMATION TECHNOLOGY

Session 2016-2017

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

RCC INSTITUTE OF INFORMATION TECHNOLOGY

[Affiliated to West Bengal University of Technology]

CANAL SOUTH ROAD, BELIAGHATA, KOLKATA-700015

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
RCC INSTITUTE OF INFORMATION TECHNOLOGY

TO WHOM IT MAY CONCERN

I hereby recommend that the Project entitled**Identification of Event in Human Object**prepared under my supervision by **Arindam Kundu , Prolay Kumar Saha , Saikat Mondal , Srijit Roy Chowdhury** (Reg. No. 131170110015,131170110052,131170110063,14117012008) Class Roll No. – CSE/2013/001 , CSE/2013/002 , CSE/2013/003 , CSE/2014/L-11) of B.Tech (7^{th} /8^{th} Semester), may be accepted in partial fulfillment for the degree of **Bachelor of Technology in Computer Science & Engineering** under West Bengal University of Technology (WBUT).

.

.....

Project Supervisor
Department of Computer Science and Engineering
RCC Institute of Information Technology

Countersigned:

.....

Head
Department of Computer Sc. & Engg,
RCC Institute of Information Technology
Kolkata – 700015.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

RCC INSTITUTE OF INFORMATION TECHNOLOGY

CERTIFICATE OF APPROVAL

The foregoing Project is hereby accepted as a credible study of an engineering subject carried out and presented in a manner satisfactory to warrant its acceptance as a prerequisite to the degree for which it has been submitted. It is understood that by this approval the undersigned do not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn therein, but approve the project only for the purpose for which it is submitted.

FINAL EXAMINATION FOR
EVALUATION OF PROJECT

1. _____

2. _____

(Signature of Examiners)

ACKNOWLEDGEMENT

We take this opportunity to express our profound gratitude and deep regards to our faculty Dr. Anup Kumar Kolya for his exemplary guidance, monitoring and constant encouragement throughout the course of this project. The blessing, help and guidance given by him time to time shall carry us a long way in the journey of life on which I am about to embark.

We are obliged to our project team members for the valuable information provided by them in their respective fields. We are grateful for their cooperation during the period of our assignment.

……..

……..

……..

……..

Table of Contents

Abstract

A long-standing goal in the field of artificial intelligence is to develop agents that can perceive and understand the rich visual world around us and who can communicate with us about it in natural language. Significant strides have been made towards this goal over the last few years due to simultaneous advances in computing infrastructure, data gathering and algorithms. The progress has been especially rapid in visual recognition, where computers can now classify images into categories with a performance that rivals that of humans, or even surpasses it in some cases such as classifying breeds of dogs. However, despite much encouraging progress, most of the advances in visual recognition still take place in the context of assigning one or a few discrete levels to an image (e.g. person, boat, keyboard etc.).

In this project we have used a model that can describe various parts of an image in natural language, from this sentences we have extracted action verbs which helps us identifying the event in which an object is busy. In particular, first we introduce a model that embeds both images and sentences into a common multi-model embedding space. This space then allows us to identify images that depict an arbitrary sentence description and conversely, we can identify sentences that describe an image. This model can take an image and directly generate a sentence description without being constrained a finite collection of human-written sentences to choose from. At the end to identify the events, we have used a parser to extract the action verbs.

1. Introduction

Advancement of computer science has made a huge progress in the area of information extraction from various documents. This document can be an image or a simple text file. However, nowadays identification of text information from document is not so difficult where as identification of event word from a human object is very difficult task. Attempt to extract temporal information is a popular and interesting research area of Natural Language Processing (NLP). Generally, events are described in different newspaper texts, stories and other important documents where events happen in time and temporal location and ordering of these events are specified. This is also important in a wide range of NLP applications that include temporal question answering, machine translation and document summarisation.

TimeML view of events as situations that *happen or occur,* or elements describing *states or circumstances* in which something obtains or holds the truth. These events are generally expressed by tensed or un-tensed verbs, normalisation, adjectives, predictive clauses or prepositional phrases.

In our present task, we have introduced about event identification from images which are also an important source of information, specially we can collect lots of information from a single image. Every image has a story behind it. We have used the Dense Captioning task, which requires a computer vision system to both localise and describe salient regions in images in natural language.

The dense captioning task generalises object detection when the descriptions consists of a single word, and Image Captioning when one predicted region covers the full image.

2. Literature Survey

Temporal relation identification based on machine learning approaches was first introduced in (Boguraev et al., 2005), (Main et al., 2006), (Chambers et al., 2007). Most of these works tried to improve classification accuracies through feature engineering. The performance of any machine learning based system is often limited by the amount of available training data. (Mani et al., 2006) introduced a temporal resigning component that greatly expands the available training data. However, this has two shortcomings, namely feature vector duplication caused by the data normalisation process and the unrealistic evaluation scheme. The solutions to these issues are briefly described in Mani et al. (2007). The main challenges involved in this test were first addressed during TempEval-1 in 2007 (Verhagen et al., 2007). This was an initial evaluation exercise based on three limited tasks that were considered realistic both from the perspective of assembling resources for development and testing and from the perspective of developing systems capable of addressing the tasks. In TempEval 2007, following types of event-time temporal relations were considered: **Task A** (relation between the events and times within the same sentence), **Task B** (relation between events and document creation time) and **Task C** (relation between verb events in adjacent sentences).
In 2010, TempEval-2, event extraction task was introduced as task B. Event extraction system was based on Support Vector Machine (SVM) approach and used various features extracted from the TimeML corpus where knowledge of semantic role labelling, WordNet and several heuristics were incorporated. Then, (Anup Kolya et al., 2011) used CRF model for event extraction where following featured were taken (i) Morphological Features (ii) Syntactic Feature (iii) WordNet Feature (iv) Semantic Roles (v) Object information of Dependency Relation(DR).

(Andrej Karpathy et al., 2016) have used the Dense Captioning task, which requires a computer vision system to both localise and describe salient regions in images in natural language. The dense captioning task generalises object detection when the descriptions consists of a single word, and Image Captioning when one predicted region covers the full image. To address the localisation and description task jointly they have proposed a Fully Convolutional Localisation Network (FCLN) architecture that processes an image with a single, efficient forward pass, requires no external regions proposals, and can be trained end-to-end with a single round of optimisation. The architecture is composed of Convolutional Network, a novel dense localisation layer, and Recurrent Neural Network language model that generates the label sequences. They have evaluatedtheir network on the Visual Genome dataset, which comprises 94,000 images and 4,100,000 region-grounded captions. They have observed both speed and accuracy improvements over baselines based on current state of the art approaches in both generation and retrieval settings.

They selected two orthogonal directions: First, rapid progress in object detection has identified models that efficiently identify and label multiple salient regions of an image. Second, recent advances in image captioning have expanded the complexity of the label space from a fixed set of categories to sequence of words able to express significantly richer concepts. However, despite encouraging progress along the label density and label complexity axes, these two directions have remained separate. In their work, they took a step towards unifying these two inter-connected tasks into one joint framework. First, they introduced the dense captioning task to predict a set of descriptions across regions of an image. Object detection is hence recovered as a special case when the target labels consist of one word, and image captioning is recovered when all images consist of one region that spans the full image. Additionally, they used a Fully Convolutional Localisation Network (FCLN) for the dense captioning task which was composed of a Convolutional Neural Network and a Recurrent Neutral Network language model.

They also introduced the task of object detection, image captioning, and soft spatial attention that allows downstream processing of particular regions in the image.

Object Detection: core visual processing module is Convolutional Neural Network (CNN) which has emerged as a powerful model for visual recognition tasks. The first application of these models to dense prediction tasks was introduced in R-CNN, where each region of interest was processed independently. In the next step, they focused on processing all regions with only single forward pass of the CNN, and on eliminating explicit region proposal methods by directly predicting the bounding boxes either in the image coordinate system, or in a fully convolutional and hence position-invariant settings. Additionally, they replaced the region of interest (RoI) pooling mechanism with a differentiable, spatial soft attention mechanism. In particular, this change allowed them the back-propagate through the region proposal network and train the whole model jointly.

Image Captioning: Several pioneering approaches have explored the task of describing images with natural language. More recent approaches based on neural networks have adopted Recurrent Neural Networks (RNNs) as the core architectural element for generating captions. These models have previously Benn used in language modelling, where they are known to learn powerful long-time interactions. Several recent approaches to Image Captioning rely on combination of RNN language model conditioned on image information, possibly with soft attention mechanisms. Similar work, **(Karpathy and Fei-Fei, year)** run an image captioning model on regions but they do not tackle the joint task of detection of description in one model. This model is end-to-end and designed in such way that the prediction for each region is a function of the global image context, which can show ultimately stronger performance. Finally, the metrics developed for dense captioning task are inspired by metrics develop for image captioning.

3.Proposed Work

The goal is to design an architecture that jointly localises region of interest and then describes each with natural language. The primary challenge is to develop a model that supports end-to-end training

with a single step of optimization, and both efficient and effective inference. Proposed architecture draws on architectural elements present in recent work on object detection, image captioning and soft spatial attention to simultaneously address these design constraints. Next we first describe the components of our model. Then we address the loss function and the details of training and interface.

3.1 Model Architecture:

3.1.1 Convolutional Network:

We use the VGG-16 architecture for its state-of-the-art performance. It consists of 13 layers of 3 cross 3 convolutions interspersed with 5 layers of 2 cross 2 max pooling. We remove the final pooling layer, so an input image of shape 3 cross W cross H gives rise to a tensor of features of shape C cross W' cross H' where C = 512, W' = [W/16], and H' = [H/16]. The output of this network encodes the appearance of the image at a set of uniformly sampled image locations, and forms the input to the localisation layer.

3.1.2 Fully Convolutional Localisation Layer:

The localisation layer receives an input tensor of activations, identifies spatial regions of interest and smoothly extracts a fixed-sized representation from each region. Out approach is based on that of Faster R-CNN, but we replace their RoI pooling mechanism with bilinear interpolation, allowing our model to propagate gradients backward through the coordinates of predicted regions. This modification opens up the possibility of predicting affine or morphed region proposals instead of bounding boxes, but we leave these extensions to future work.

3.1.3 Inputs/Outputs:

The localisation layer accepts a tensor of activations of size C cross W' cross H'. It then internally selects B regions of interest and returns three output tensors giving information about these regions:

A. **Region Coordinates:** A matrix of shape B cross 4 giving bounding box coordinates for each output region.
B. **Region Scores:** A vector of length B giving a confidence score for each output region. Regions with high confidence scores are more likely to correspond to ground-truth regions of interest.
C. **Region Features:** A tensor of shape B cross C cross X cross Y giving features for output regions; is represented by an X cross Y grid of C-dimensional features.

3.1.4 Convolutional Anchors:

Similar to Faster R-CCN, localisation layer predicts region proposals by regressing offsets from a set of translation-invariant anchors. In particular, we project each point in the W' cross H' grid of input features back into the W cross H image plane, and consider

k anchor boxes of different aspect ratios entered at this projected point. For each of these k anchor boxes, the localisation layer predicts a confidence score and four scalars regressing from the anchor to predicted box coordinates. These are computed by passing the input feature map through a 3 cross E convolution with 256 filters, a rectified linear nonlinearity, and a 1 cross 1 convolution with 5k filters. This results in a tensor of shape 5k cross W' cross H' containing scores and offsets for all anchors.

3.1.5 Box Regression: By adopting the parameterisation to regress from anchors to region proposals. Given an anchor box with centre (x_a, y_a), width w_a, and height h_a, model predicts scalars (t_x,t_y, t_w, t_h)giving normalised offsets and log-space scaling transforms, so that the output region has centre (x, y) and shape (w, h)given by

$$x = x_a + t_x w_a \qquad y = y_a + t_y h_a \qquad (1)$$

$$w = w_a \exp(t_w) \qquad h = h_a \exp(h_w) \qquad (2)$$

Note that the boxes are discouraged from drifting too far from their anchors due to L2 regularisation.

3.1.6 Box Sampling: Processing a typical image of size W = 720, H = 540 with k = 12 anchor boxes gives rise to 17,280 region proposals. Since running the recognition network and the language model for all proposals would be prohibitively expensive, it is necessary to subsample them.

At training time, we sample a mini batch containing B = 256 boxes with at most B/2 positive regions and the rest negatives. A region is positive if it has an intersection over union (IoU) of at least 0.7 with some ground-truth region; in addition, the predicted region of maximal IoU with each ground-truth region is positive. A region is negative if it has IoU < 0.3 with all ground-truth regions. Out sampled mini batch contains $B_P <= B/2$ positive regions and $B_N = B - B_P$ negative regions, sampled uniformly without replacement from the set of all positive and all negative regions respectively.

At test time we subsample using greedy non-maximum suppression (NMS) based on the predicted proposal confidences to select the B = 300 most confident proposals. The coordinates and confidences of the sampled proposals are collected into tensors for shape B cross 4 and B respectively, and are output forms the localisation layer.

3.1.7 Bilinear Interpolation: After sampling, we are left with region proposals of varying sizes and aspect ratios. In order to interface with the full-connected recognition network and the RNN language model, we must extract a fixed-size feature representation for each variably sized region proposal.

Fast R-CNN proposes an RoI pooling layer where each region proposal is projected onto the W' cross H' grid of convolutional features and divided into a coarse X cross Y grid aligned to pixel

boundaries by rounding. Features are max-pooled within each grid cell, resulting in an X cross Y grid of output features.

The RoI pooling layer is a function of two inputs: convolutional features and region proposal coordinates. Gradients can be propagated backward from the output features to the input features, but not to the input proposal coordinates. To overcome this limitation, replacement of RoI pooling layer with bilinear interpolation can be effective.

Concretely, given an input feature map U to shape C cross W' cross H' and a region proposal, we interpolated the features of U to produce an output feature map V of shape C cross X cross Y. After projecting the region proposal onto U and compute a sampling grid G of shape X cross Y cross 2 associating each element of V with real-valued coordinates into U. If $G_{i,j} = (x_{i,j} , y_{i,j})$ then $V_{c,i,j}$ should be equal to U at $(c, x_{i,j} , y_{i,j})$; however since $(x_{i,j} , y_{i,j})$ are real-valued, we convolve with a sampling kernel k and set

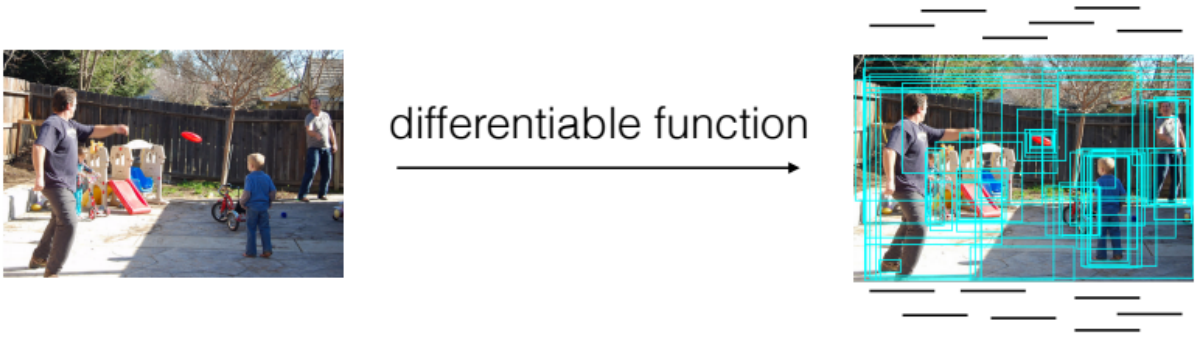
$$V_{c,i,j} = \Sigma^{w}_{i'=1} \Sigma^{H}_{j'=1} U_{c,i',j'} k(i' - x_{i,j}) k(j' - y_{i,j}) \qquad (3)$$

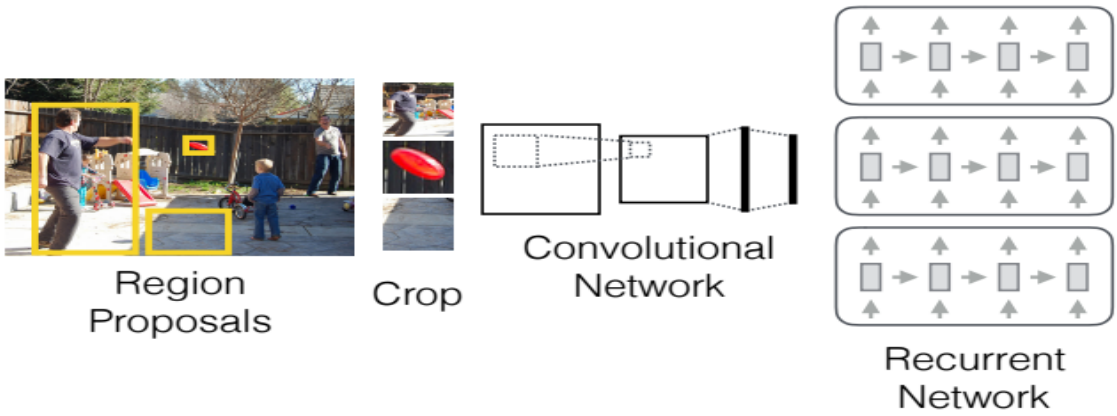
Bilinear sampling, corresponding to the kernel k(d) = max (0,1 - |d|) is used. The sampling grid is a linear function of the proposal coordinates, so gradients can be propagated backward into predicted region proposal coordinates. Running bilinear interpolation to extract features for all sampled regions gives a tensor of shape B cross C cross X cross Y, forming the final output from the localisation layer.
After getting the image from dense cap model with region caption, we are using Stanford CoreNLP parser to find action verbs from it. In parser the whole region caption is given as input and after parsing, verbs can be extracted from it.

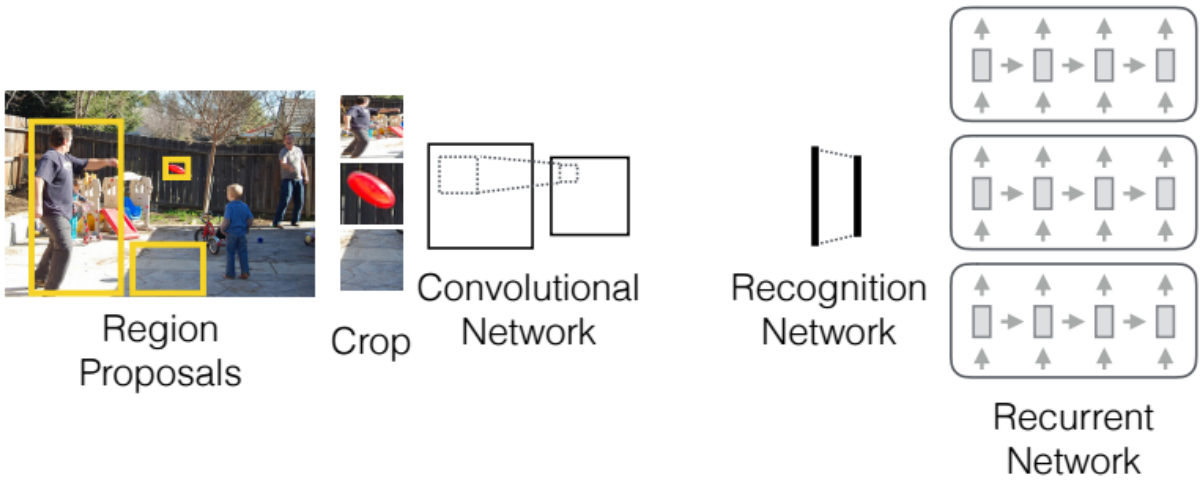
4. System Design

End to End Learning :

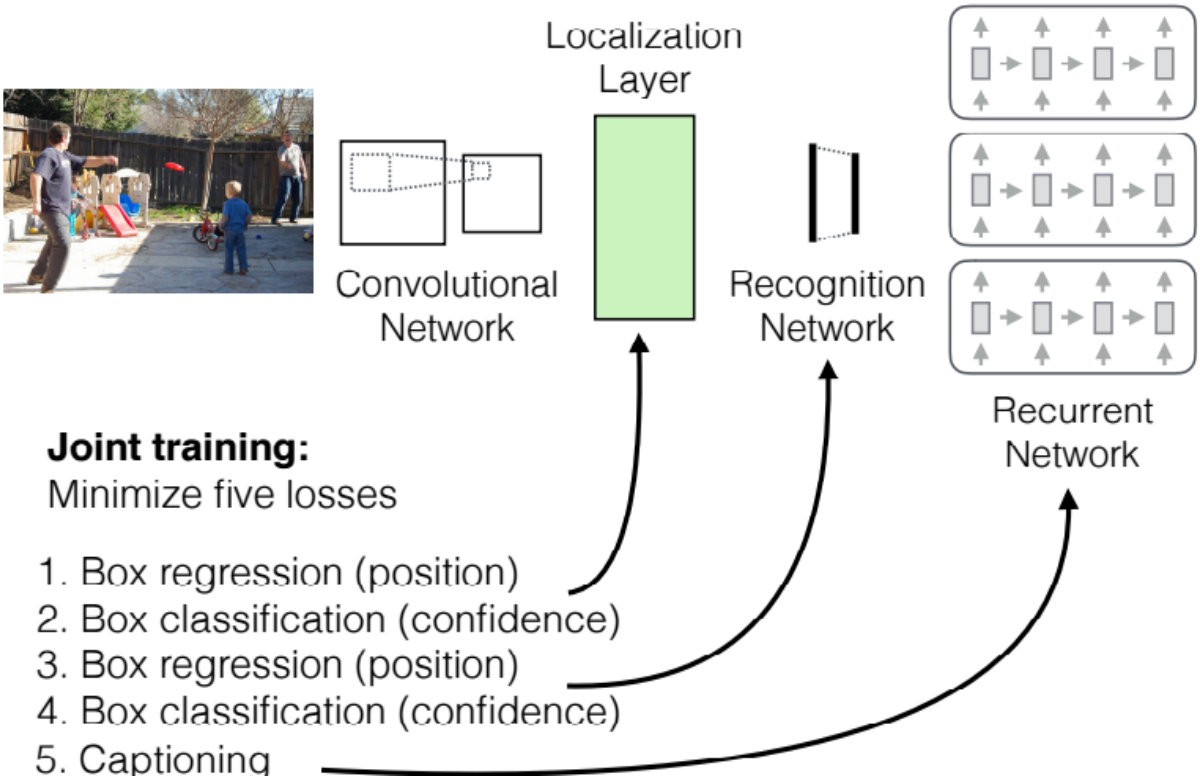
Formulate a single differentiable function from inputs to outputs



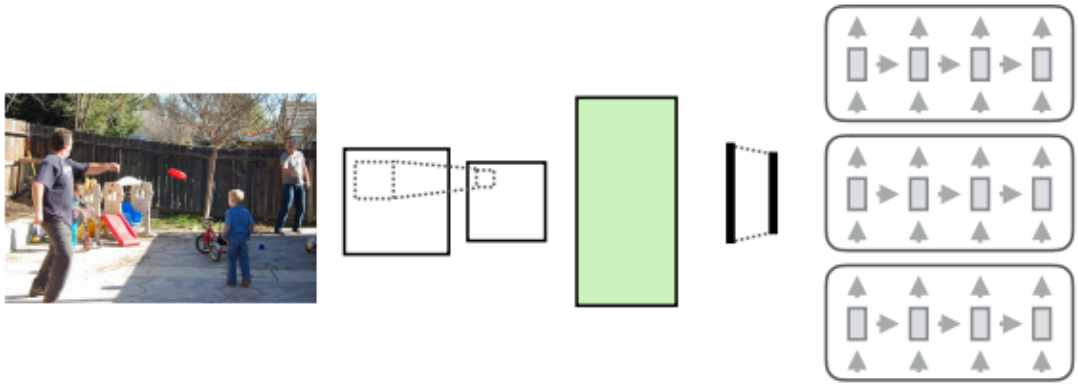




Dense Captioning Architecture



Advantages over Prior Work



- **No Context** : The Prediction does not include context outside if each region .

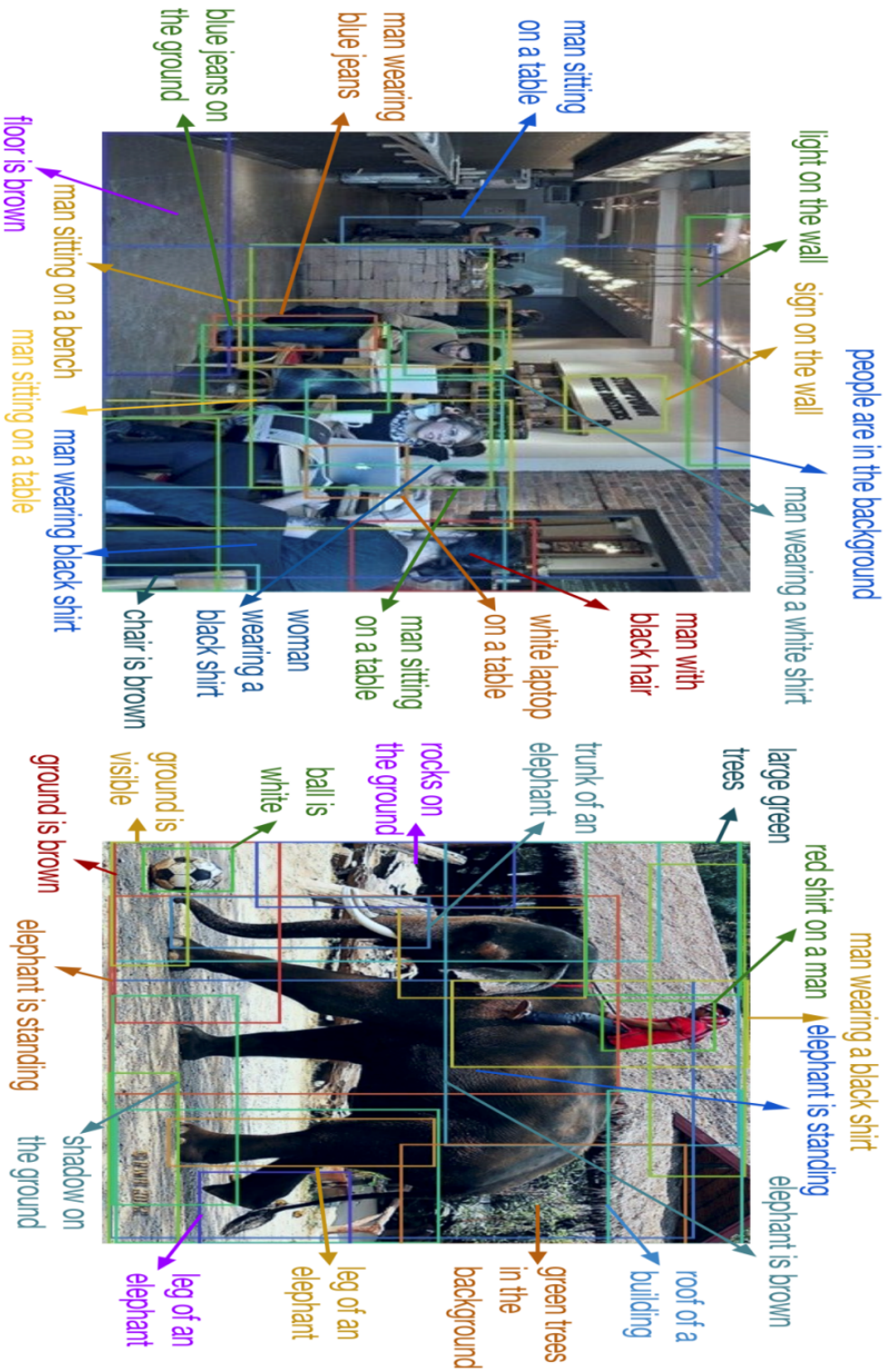
  Solved :Features far up the CNN are the function of much larger regions.

- **Inefficient :**We must forward every region independently .

  Solved : Reuse computation alreadydone inside the ConvNet.

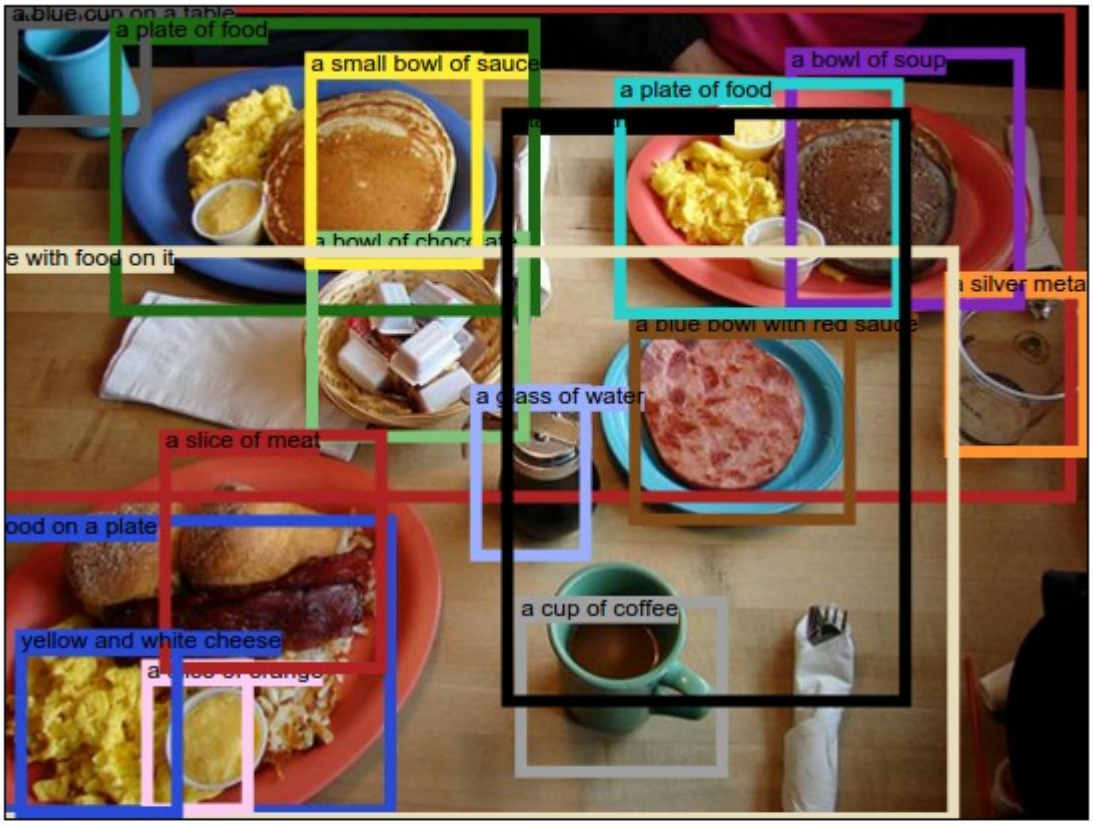
- **Not end-to-end :** The use of external region proposals .
- **Solved : Can be trained end to end .**

OUTPUT:

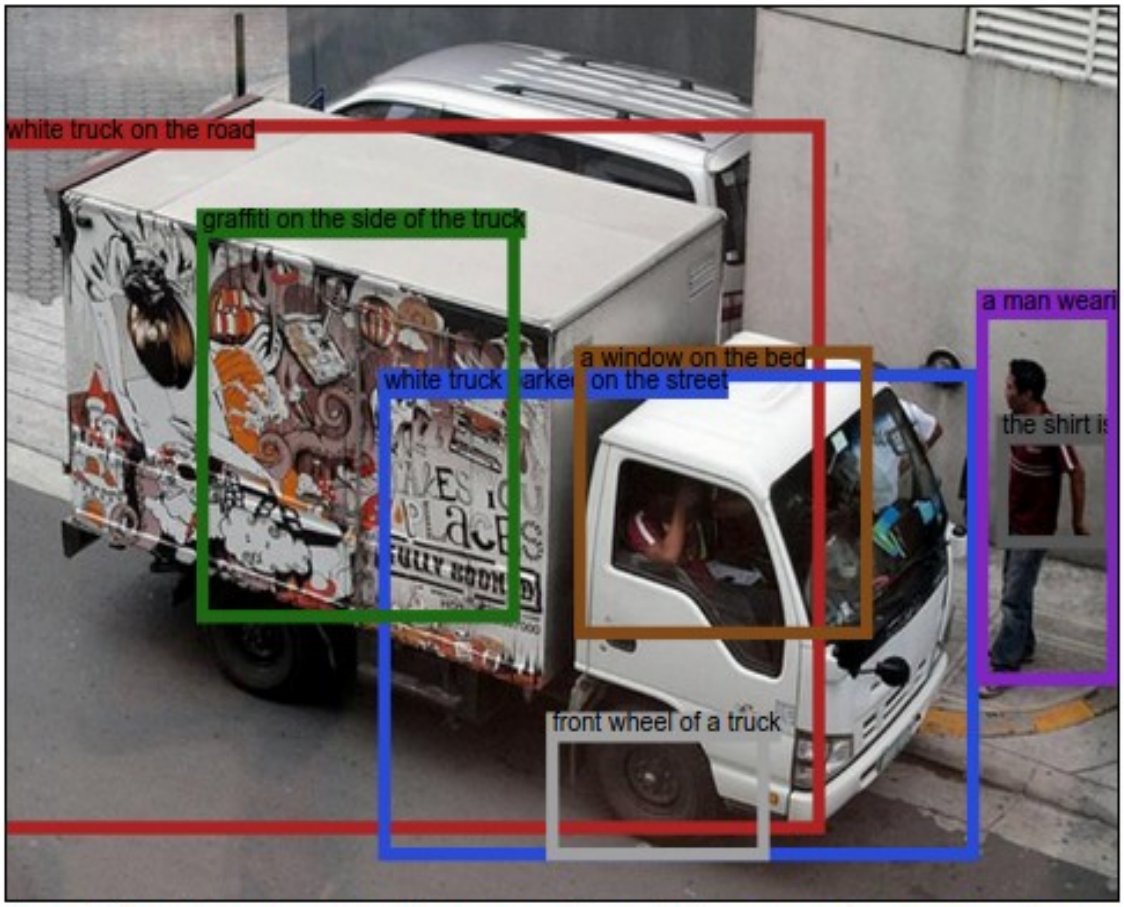

people are in the background
light on the wall
sign on the wall
man wearing a white shirt
man with black hair
white laptop on a table
man sitting on a table
woman wearing a black shirt
chair is brown
man wearing black shirt
man sitting on a table
man sitting on a bench
floor is brown
blue jeans on the ground
man wearing blue jeans
man sitting on a table
man wearing a black shirt
elephant is standing
elephant is brown
red shirt on a man
large green trees
roof of a building
green trees in the background
leg of an elephant
leg of an elephant
shadow on the ground
elephant is standing
ground is brown
ground is visible
ball is white
rocks on the ground
trunk of an elephant

5. Some Examples

bus parked on the street. a city street scene. front windshield of a bus. man walking on sidewalk. a silver car parked on the street. a city scene. a green traffic light. a building in the background. the bus has a number. a large building. a brick building. red brick building with windows. a blue sign with a white arrow. white lines on the road.



a plate of food. food on a plate. a blue cup on a table. a plate of food. a blue bowl with red sauce. a bowl of soup. a cup of coffee. a bowl of chocolate. a glass of water. a plate of food. a silver metal container. a small bowl of sauce. table with food on it. a slice of orange. a table with food on it. a slice of meat. yellow and white cheese.



a white truck on the road. white truck parked on the street. the shirt is red. graffiti on the side of the truck. a window on the bed. a man wearing a black shirt. front wheel of a truck.

6. Algorithm Implemented

Step 1: Getting an image id

First, let's get an image from the dataset.

Code segment:

```
importmatplotlib.pyplotasplt
frommatplotlib.patchesimportRectangle
fromsrcimportapiasvg
fromPILimportImageasPIL_Image
importrequests
fromStringIOimportStringIO
```

```
ids=vg.GetImageIdsInRange(startIndex=0,endIndex=1)
image_id=ids[0]
print"We got an image with id: %d"%image_id

%matplotlibinline
```

We got an image with id: 1

Step 2: Getting the image data

Next, we will get some data about the image. We specifically want to know the image's url.

```
image=vg.GetImageData(id=image_id)
print"The url of the image is: %s"%image.url
```

The url of the image is: https://cs.stanford.edu/people/rak248/VG_100K_2/1.jpg

Step 3: Getting the region descriptions

Now, let's get all the region descriptions for this image.

```
regions=vg.GetRegionDescriptionsOfImage(id=image_id)
print"The first region descriptions is: %s"%regions[0].phrase
print"It is located in a bounding box specified by x:%d, y:%d,
width:%d, height:%d"%
(regions[0].x,regions[0].y,regions[0].width,regions[0].height)
```

The first region descriptions is:

The clock is green in colour. It is located in a bounding box specified by x:421, y:57, width:82, height:139

Step 4: Visualizing some regions

Now, we will visualize some of the regions. The x, y coordinates of a region refer to top left corner of the region. Since there are many regions, we will only visualize the first 8.

```
fig=plt.gcf()
fig.set_size_inches(18.5,10.5)
defvisualize_regions(image,regions):
response=requests.get(image.url)
img=PIL_Image.open(StringIO(response.content))
plt.imshow(img)
ax=plt.gca()
forregioninregions:
ax.add_patch(Rectangle((region.x,region.y),
region.width,
region.height,
fill=False,
edgecolor='red',
linewidth=3))
ax.text(region.x,region.y,region.phrase,style='italic',bbox={'faceco
lor':'white','alpha':0.7,'pad':10})
fig=plt.gcf()
plt.tick_params(labelbottom='off',labelleft='off')
plt.show()
visualize_regions(image,regions[:8])
```

7. Implementation Details

DenseCap is implemented in Torch, and depends on the following packages :

torch/torch7 , torch/nn , torch/nngraph , torch/image , lua-cjson , qassemoquab/ stnbhwd , jcjohnson/torch-rnn

After installing torch , we can install/update these dependencies by running the following :

- **luarocks install torch**
- **luarocks install nn**
- **luarocks install image**
- **luarocks install lua-cjson**
- **luarocks install https://raw.githubusercontent.com/qassemoquab/stnbhwd/master/stnbhwd-scm-1.rockspec**
- **luarocks install https://raw.githubusercontent.com/jcjohnson/torch-rnn/master/torch-rnn-scm-1.rockspec**

GPU Acceleration

If we have a NVIDIA GPU and want to accelerate the model with CUDA, we need to install torch/cutorch and torch/cunn; we can install / update these by running:

- **luarocks install cutorch**
- **luarocks install cunn**

If we want to use NVIDIA's cuDNNlibrary , we need to register for the CUDA Developer Program (it's free) and download the library from NVIDIA's website; we also need to install the cuDNN bindings for Torch by running :

- **luarocks install cudnn**

7.1 Pre-trained Model

We can download a pretrained DenseCap model by running the following script:

- **sh scripts/download_pretrained_model.sh**

This will download a zipped version of the model (about 1.1 GB) to data/models/densecap/densecap-pretrained-vgg16.t7.zip, unpack it to data/models/densecap/densecap-pretrained-vgg16.t7 (about 1.2 GB) then delete the zipped version.

This is not the exact model that was used in the paper, but is has comparable performance; using 1000 region proposals per image, it achieves a mAP of 5.70 on the test set which is slightly better than the mAP of 5.39 that we report in the paper.

7.2 Running New Images

To run the model on new images,we use the script **run_model.lua**. To run the pretrained model on the provided elephant.jpgimage, use the following command:

- **thrun_model.lua -input_imageimgs/elephant.jpg**

By default this will run in GPU mode; to run in CPU only mode, simply add the flag **-gpu -1.**This command will write results into the folder vis/data. We have provided a web-based visualizer to view these results; to use it, change to the vis directory and start a local HTTP server:

- **cd vis**
- **python -m SimpleHTTPServer 8181**

We will now point our web browser to **http://localhost:8181/view_results.html.**

To run an entiredirectory of images on which we want to run the model, we use the **-input_dir** flag instead:

- **thrun_model.lua -input_dir /path/to/my/image/folder**

7.3 Running Parser

The output of DenseCap model is given as input to the Stanford CoreNLP parser. From this region mapping input, parser identifies the verbs to detect event in object. We are using an off line version of this parser which is mostly a program, coded using java language. The Stanford POS Tagger is designed to be used from the command line or programmatically via its API. There is a GUI interface, but it is for demonstration purposes only; most features of the tagger can only be accessed via the command line. To run the demonstration GUI you should be able to use any of the following 2 methods:

1) java -mx200m -classpathstanford-postagger.jar:lib/* edu.stanford.nlp.tagger.maxent.MaxentTaggerGUI models/wsj-0-18-left3words-distsim.tagger

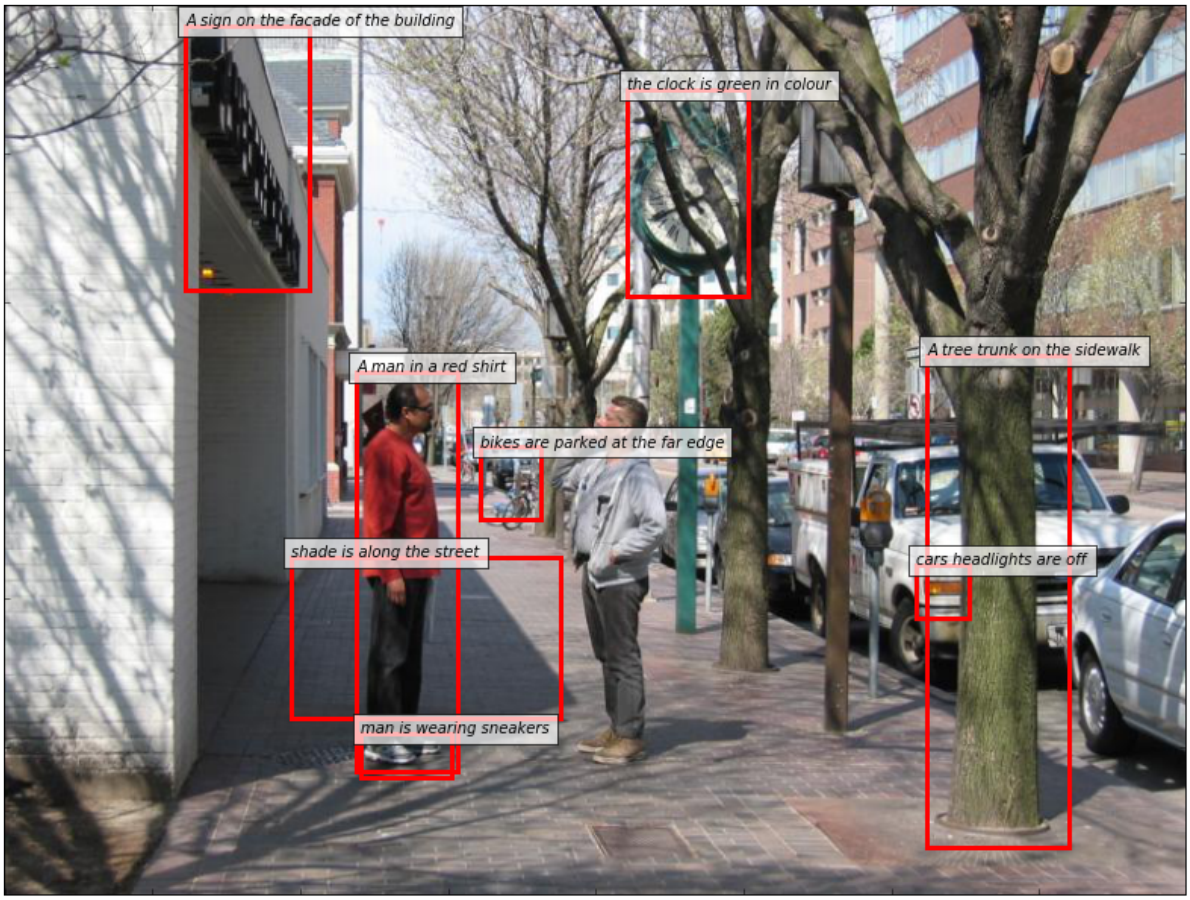
2) Running the appropriate script for your operating system:

stanford-postagger-gui.bat

./stanford-postagger-gui.sh

8. Output:

output from densecap model:



input to Stanford CoreNLP parser:

A sign on the facade of building, the clock is green in colour, a man in a red shirt, bikes are parked at the far edge, shade ins along the street, man is wearing sneakers, a tree trunk on the sidewalk, cars headlights are off

Stanford CoreNLP parser output:

```
1 A/DT sign/NN on/IN the/DT facade/NN of/IN building/NN ,/, the/DT clock/NN is/VBZ green/JJ in/IN colour/NN ,/, a/DT man/NN in/IN a/DT red/JJ shirt/NN ,/, bikes/NNS are/VBP
  parked/VBN at/IN the/DT far/JJ edge/NN ,/, shade/NN ins/NNS along/IN the/DT street/NN ,/, man/NN is/VBZ wearing/VBG sneakers/NNS ,/, a/DT tree/NN trunk/NN on/IN the/DT
  sidewalk/NN ,/, cars/NNS headlights/NNS are/VBP off/IN
```

Extracted events are: parked, wearing,

9. Limitation

To run this model we have to faced lots of problem and failure. After detail diagnosis, we have identified that machine specification is an important area of concern. Initially, we were using a machine with i5 processor and 4gb of RAM. Processing single image is taking a long time but if the image size increases 100kb, the machine is not showing output. Sometimes terminal shows out of memory error directly. Then we have come to know that a machine with i5 processor and 8gb of RAM would be more reliable for running this model. This machine can process a bit large image without any critical issue. We can also run more than one image smoothly. The accuracy level of this model can be improved by training this model with large dataset and region description. Some dataset that we have identified for training dense-cap model, are of size 9 to 12 GB. Huge dataset like the above one also demands premium machine configuration.

10. Future Work & Conclusion

Starting from literature survey to output we have studied many published paper to understand the process of extraction of the event from the input image. We have used DenseCap model for the image to natural language text output and by using Stanford CoreNLP parser, we have extracted events from natural language. Our next plan is to use a different dataset to train this model for the improvement of accuracy. Visual Genome has provided huge dataset, region description and image metadata that can be used to train this model. In future, we shall integrate the parser and the model to create a new integrated system for event extraction. For all this future progress, the main obstacle is to understand the model architecture and development above it. Though machine limitation is also a major factor. Till now we have faced a lot of machine related problem and a significant decrease in machine performance in region captioning process. Large dataset requires premium machine configuration which is not easily available. We are hopeful that in remaining semester we shall achieve our final goal by completion of small checkpoints.

11. References

1. https://visualgenome.org/api/v0/api_beginners_tutorial.html
2. https://github.com/jcjohnson/densecap
3. https://visualgenome.org/api/v0/api_home.html
4. http://cs.stanford.edu/people/karpathy/densecap/
5. https://www.youtube.com/?gl=IN
6. https://www.youtube.com/channel/UClKO7be7O9cUGL94PHnAeOA
7. https://www.youtube.com/channel/UCMoXOGX9mgrYNEwpcIQUcag
8. A first step toward evaluating events, Time expressions and temporal relations by Anup Kr. Kolya, Asif Ekbal and Sivaji Bandyopadhyay
9. A hybrid approach for event extraction and event actor identification by Anup Kr. Kolya, Asig Ekbal and Sivaji Bandyopadhyay
10. A crf based approach of annotation of temporal expression, event and temporal relations by Anup Kr. Kolya, Amitava Kundu,Asig Ekbal, Sivaji Bandyopadhyay and Rajdeep Gupta
11. Connecting images and natural language by Andrej Karpathy
12. DenseCap: Fully Convolutional Localization Networks for Dense Captioning by Justin Johnson, Andrej Karpathy and Li Fei Fei
13. http://nlp.stanford.edu:8080/corenlp/ -- for parsing event.
14. http://nlp.stanford.edu/software/tagger.shtml -- Standford Log-linear Part-Of-Speech Tagger.