Knowlywood: Create Activity Knowledge Base

Chirag Shah
Arizona State University
crshah2@asu.edu

Prasanth Kumar Dhanamjayulu
Arizona State University
pdhanamj@asu.edu

Tianyang Chen
Arizona State University
tchen61@asu.edu

Divyesh Dnyanmothe
Arizona State University
ddnyanmo@asu.edu

Abstract

Given movie scripts, the goal is to create an activity knowledge base which will contain values for each of these parameters: Participant, Previous activities, Next activities, similar activities and Source. We have created a knowledge graph structure where in each activity has a node and the each node has an edge with the previous activity and next activity.

1 Problem Definition and Motivation

In this problem, we will be provided with a set of top movie scripts from popular datasets. The movie script will be a readable text-based script. An information extraction pipeline for capturing activities with their participating agents through semantic parsing is used along with ordering of the activities sequentially to form a knowledge base. The motivation of the problem is interpreting user intentions in natural language querying and also improving the understanding of visual contents in photos and videos.

2 Techniques used

Movie scripts are typically available in plain text format and use similar structure. In this problem, we use a customized semantic parsing pipeline that starts with the raw input scripts, performs information extraction and disambiguates constituents to construct a frame structure for candidate activities.

Each input sentence is parsed. We pass the parsed trees to a rule defined function to get an activity and its previous and next activity. We have an assumption that the previous activity can possibly be in the previous same or one of the two statements and similarly the next activity can be in the same statement or one of the two next statements. After this if a sentence has an activity we search for participants for that activity in that particular sentence. We then map these activities to their corresponding participants. We parse the movie scripts one after the other to get an activity file which stores all the information regarding the activity.

The information gathered in previous step is then merged on the basis of similarity between activity names using Classifier4j.

We then find the semantic similarities between the test activity phrases and then merge activities gathered from the previous step. We then merge all the information from top few similar activities to create a training model for the HMM. A dictionary of categories based on the EOWL (English Open Word List) list of words is used for the same. The training file for an HMM contains tuples where each tuple represents a set of similar activity.

The Hidden Markov Model gives us the list of the most probable three previous activities and two next activities. We select 5 participants and similar activities given a particular test activity.

3 Existing software resources that can be used

We have used OpenNLP for tokenization, POS-tagging, parsing and chunking. The sentence splitter and POS parser are in the OpenNLP toolkit.

Classifier4J is a java library that provides an API for automatic classification of text. An enhancement of the Vector-Space analysis found within the Classifier4j is used for this purpose. A dictionary of categories based

on the EOWL (English Open Word List) list of words is used for the same.
We have used JAVA SWING for creating the GUI which has simple textbox, data table in which all the data is displayed.

4 Activity Extraction

We first define initial frames that we need to fill to store the activity information. The data structure is:

```
Class Activity
{
        String name;

        ArrayList<Activity> prev, next;

        ArrayList<String> participant;
}
```

Each Activity is a node in the whole graph. Each Activity has a list of previous activities and a list of next activities. If activity A is activity B's next activity, then activity B is the previous activity of activity A, and we assume they are connected in the graph.
Once we have the Activity class, we can start to extracting activities from movie scripts.
We first use a lot of examples to test which POS rules would work well in our program. We select some of them to form our POS rule set, which is used to parse the script and extract activities from it. Meanwhile, we also form another POS rule set to extract participant of those activities.
The extracted activities, has stopwords and other unnecessary characters..In order to remove those stopwords, we define a function removeStopword(), it can remove all the symbols at the end of the activity and remove some other characters. like "…", ";",
We use a HashMap<String, Activity> and an ArrayList<Activity> to store all the activities we extracted. The key of the map is the name of the Activity, and the value the Activity itself. In this way, we can get the Activity object from map just using its name. ArrayList is helpful for accessing the Activity sequentially, and HashMap is helpful for accessing the Activity randomly. Every time we extract an activity from the script, we first check if there is a same activity existed in the map, if there is not, we need to create a new Activity and add it to the map.
For each activity, we assume activities in the previous three sentences are the previous activities, and activities in the next three sentences are the next activities. So, when we extract an activity A, we have to check all activities in the previous three sentences, add them to A's previous activity set and add A to their next activity set. Connecting all these activities will result our initial knowledge graph.
After the whole script is parsed, we have formed a list of all the activities. We store all these activity information into a text file, for each activity, we store the activity name, previous activities, next activities and participants.

.

5 Resolving similarity disambiguation

The information gathered in previous step is then merged on the basis of similarity between activity names using Classifier4j. A similarity measure is calculated for any two activities and this measure usually ranges from 0 to 1. A threshold of 0.5 is used in our effort to merge activities. The previous activities, next activities and participants of similar activities are merged together to form the information for an activity. Thus duplicate activities are eliminated in the process of merging activities based on similarity measure and still preserving the information of duplicate activities by combining them. An enhancement of the Vector-Space analysis found within the Classifier4j is used for this purpose. This enhancement includes a new dictionary of categories based on the EOWL (English Open Word List) list of words and the categories for each word are calculated based on DISCO's semantics. DISCO (extracting distributionally related words using coreference) allows to retrieve the semantic similarity between arbitrary words and phrases. The similarities are based on the statistical collection of very large text collections. The vector space analysis of the Classifier4j doesn't work well with short sentences while this enhancement does. Also, the vector space analysis does not take into account the semantic meanings of words while this enhancement does as it uses DISCO's semantics for words.

We assign the following tags to an activity information for a given activity while writing to the file.

part->Participants
next->Next Activities
prev->Previous Activities

6 Training Hidden Markov Model

We train the Hidden Markov Model by feeding the similar activity files parsed above. We then read the HMM Tuple one by one separating them by a pre defined Splitter. We create a word tag count map wherein the tag is the previous or next state. We then read the tuples one by one and then increase the count if we already have the same word for a given tag. Once the entire train file is read we compute the probability of all the words given a tag. We thus get the probability of a given word in that tag state. Once we finish training we then sort all the observations in next and previous

state on the basis of their probability in ascending order. We then use a function which gives the most probable three previous and three next activities for a given test activity state. We store all this information in this activity object. We then write this information to our knowledge base so that the next time when we enter the same activity we understand that we already have learnt information for the given activity. We also update the information in our intermediate information file wherein we set the graph structure wherein we update the respective edges for the test activity in following manner.
For all the previous activities we set the current test activity to be its next activity.
For all the next activities we set the current test activity to be its previous activity.
Given a test activity, we use the HMM model defined above to predict three most probable previous and next activities for the given activity.

We assign the following tags to an activity information for a given activity while writing to the file.

part->Participants
next->Next Activities
prev->Previous Activities
simi->Similar Activites

7 Problems faced

First, for the activities, we are getting articles like "the", "a", etc. at the end of an activity while detecting activities from the script when we create the activity file in first step. Also while detecting participants we get words which do not denote the behavior of a participant.

8 Evaluation Parameters

Manually check for activities by entering the activity and verify the values of the parameters: Participant, Previous activities, Next activities, Similar activities.

9 Validating Results

While creating the training file for a small amount of data we used a threshold of 0.25 as a similarity measure between the test activity and the activities present in file which we get after merging the detected activities. The System showed an accuracy of 40% for the information we received for a given test activity. We ran more movie scripts and increased the threshold for similarity measure to 0.60 and received an improvement up to 60% accuracy. However if the system is not able to learn any information for a given activity it fails to display any information.

10 Example

Let us consider a part of the script from the movie 17 again.
An empty gymnasium except for a shirtless MIKE O'DONNELL, 17. Mike stands feet BEYOND the 3 point line, grabs balls from a hopper and rapidly shoots, shoots, shoots. SWISH...SWISH...SWISH. This kid's automatic.

Let the activity be Grabs Ball.
We will start filling the frames one by one.
Activity -> Grabs Ball
Parent -> Grabs
Participant -> Mike (Kid)
Previous Activity -> stands feet beyond the three line
Next Activity -> shoots
Similar Activity -> Catch Ball, Fetch Ball Source -> Movie Data-Set.

Execution steps of filling the frame for an activity:

Step 1: To get the activity and related information such as previous activity, next activity and participant from movie script. Take the first two sentences before the activity and the next two sentences after the activity to get previous and next activity.
Step 2: Go through the test data activity and get similar activities from training data.
Step 3: We then use the data from the similar activities to train the hmm model.
Step 4: We get the most probable previous activity, next activity and participant for the given activity.

11 Related Works

Knowledge extraction in the form of frames of activities has been done before as seen in [1] where the scripts are parsed using line indentation and a machine learning based classification is used for describing human actions. To align the scripts with the video the time information from the movie sub-titles is used.
We have used the concept of having a time frame of two sentences before and after the activity sentence for detecting previous or next activities from [2] and [1]. The former uses a frame of 8 words. [1] and [3] uses their knowledge extract ion techniques further to state actions from corresponding images and frames.

12 Possible Challenges

Activities can have different sense in different situation. Like shoot a video or shoot with a gun. Disambiguating the activities will play an important role.

There is a possibility that the activity action can be after a certain set of lines which may not fit the frame of two lines before and after the activity sentence. So we use the same activity and try to create previous activity and next activity from more than one script.

References

[1] I.Laptev, M.Marszalek, B.Rozenfeld and C.Schmid. Detecting Human Actions from Movies, Proc. CVPR'08.

[2] R. Srikant, R. Agarwal: Mining sequential patterns: Generalizations and Performance Improvements, EDBT 1996.

[3] Niket Tandon, Gerard de Melo, Abir De, Ge rhard Weikum: Lights, Camera and Action: Knowledge Extraction from Movie Scripts, WWW 2015.