Eventspotter

spotting entities that talk about events

Project Report

Amar Kadamalakunte , Meghana Sreekanta Murthy

Fall 2012 - Spring 2013

Supervisors:
Raphaël Troncy
Giuseppe Rizzo

EURECOM
Multimedia Department

Contents

1 Abstract

By definition, an event is an occurrence in the past, present or future. Events could be classified as musical events, sports events, news events, etc. The aim of our work is to develop an approach to spot the existence of musical event entities in free form text. The Eventspotter uses a large dataset called EventMedia to identify candidate event spots in the text, and disambiguates them by assigning a confidence score. The confidence score itself is based on the identification of correlated entities (event location or artists) in the surrounding text and cosine similarity measure between the surrounding text and existing description of event in the dataset. We used precision, recall and f-measure to evaluate our approach. We used event description from the Eventmedia dataset as input and found that the Eventspotter performed better than some exisiting approaches for event detection in the musical domain. We present our method and findings in this paper.

1 Introduction

We first take a detailed look at the Eventspotter Approch and then in subsequent sections we speak at length about the golden data set creation, usage of 10-fold cross validation to evaluate the performance of the system and conclude with our thoughts on how we can potentially better the performance of the eventspotter with the help of machine learning techniques.

2 Related Work

Many people have tried to tackle the problem of entity detection in the past. While some have relied purely on the linguistic properties of text others have used context and domain knowledge to accurately detect entities. [1] was one such approach that inspired us during the literature survey phase of the Eventspotter. In this case, the utilisation of context and domain knowledge along with the application of real world constraints proved to be beneficial in improving the accuracy of entity detection. The main goal was to establish semantic relationships between entities in social networking and content sharing sites like MySpace and Youtube. In this approach, the authors used an edit- distance based spotter to match the knowledge base with the text taken from online forums discussing popular music. They then checked for domain and context information and applied an SVM classifier. Further, a study of [2] introduced us to the approach of identifying entities based on knowledge from populated ontologies such as DBWorld and DBLP. They used clues such as entity names, text-proximity, text co-ocurrence, popularity of entities, semantic relationships between entities for disambiguation. This approach further vindicated our idea of using the large Eventmedia knowledge base to detect events in Eventspotter.

As part of another approach , DBpedia URIs were used for automatic annotation of texts. The DBPedia Spotlight [3] used an extensive ontology with 272 classes, called DBpedia, and also provided the flexibility of choosing domain of interest. 2 sources of textual references include : the lexicon created from the DBPedia graph (of labels, redirects, disambiguations) and wikilinks. The Eventspotter derives various facets of the DBpedia Spotlight approach including the 4 main stages : (1)Spot phrases that may indicate a mention of a DBpedia resource, (2)Select candidates by spotted phrase-to-resources mapping, (3)Disambiguate by using the context around spotted phrase, (4)Configure the annotation by customization from user with Resource Prominence, Topic Pertinence, Contextual Ambiguitvalidy, Disambiguation Confidence.We also studied two interesting methodologies that were applied to social media and wikipedia for detection of events. TWICAL [4] focused on linguistic properties and temporal expressions to detect events, classified them using the context of surrounding phrases and

finally ranked them according to strong associations with a unique date. The Wikipedia Live Monitor[5] tracked edits on various linguistic versions of a single document on wikipedia and relied on cross-language, full-text search across social networks in order to evaluate the [2] plausability of the edits being events.

Although the above mentioned approaches are effective in entity spotting, there are a few limitations with each. In [1], the authors cite the absence of a training corpus and regions of text being commonly used words, as limiting factors. They also speak of the ambiguous nature of song titles being a major limitation. In [2] the non-exhaustive nature of any knowledge base proves to be the biggest hurdle. The same is the case with the DBpedia spotlight approach defined in [3], where the authors do not speak of using linguistic tools such as a Conditional Random Field classifier to detect entities that are non existent in the knowledge base.Both TWICAL [4] and Wikipedia Live monitor [5] are dependant on social networking sites to detect events. As a result, events that are not publicised in the social networking domain go undetected. The fact that each of these approaches have limitations of their own, goes to show that there is still scope for improvement in the field of event detection. The EventSpotter project is an attempt to meet this need for a more complete approach towards event detection.

2.2 BookSpotter

BookSpotter Enhancement Engine by Sztaki[1] takes a knowledge based approch to find book occurrences within text. It loads a list of titles into memory and matches them with tokens of input text to spot book occurrences. It further disambiguates each spot using the author string stored in a database. If one or more authors are present in the text, the spot is considered as a valid one. The BookSpotter enhancement engine is designed to work as a module inside Apache Stanbol[2]. Apache Stanbol provides a set of reusable components for semantic content management. The BookSpotter uses Stanbol for presenting the detected entities as RDF. We were greatly inspired by the idea behind the BookSpotter project. We spent a considerable amount of time analysing and understanding the logic of the BookSpotter system. In a way, the EventSpotter project stands upon the shoulders of BookSpotter. We supplement the logic of the BookSpotter with event-specific features such as presence of: artists, location, date etc.

[1] http://blog.iks-project.eu/introducing-bookspotter-enhancement-engine-by-sztaki/
[2] http://stanbol.apache.org/

3 Approach

The Eventspotter approach has four stages namely: (1)Preprocessing, (2)Candidate selection, (3)Disambiguation and (4)Postprocessing. It is imperative to restate that we have reused components of the BookSpotter which act as a foundation on which the EventSpotter is buit on.

3.1 Overview

Before we look at each of these in greater detail, it is important to first gain a high level understanding of the entire system.

[diagram]

3.1.1 EventMedia Dataset

EventMedia is an big hub obtained from three public event directories: Last.fm, Eventful and Upcoming and the media directory: Flickr. In [6], EventMedia is described to be a platform that uses semantic web technologies to aggregate and interlink real-time heterogeneous data sources. Information within Eventmedia is in the form of encapsulated media and eventdescriptions, which are supported by background knowledge from data sources such as DBpedia, MusicBrainz, BBC and Foursquare.

We retrieved a snapshot of the EventMedia database which constituted 67957 events. This snapshot, taken on 26th November 2012, contains metadata for events that occurred during the year 2012. Each event in the eventmedia snapshot has the following attributes: eventId(url to a web page corresponding to the event in data.linkedevents.org), eventTitle, publisher, date, locations, category, agent(artists/bands performing at the event) and event description. We provide in listing1 an example of a sample record from the EventMedia dataset. The events belonged to one or more of the following categories: Musical Concert, Festivals, Nightlife, Social Gathering, Performing Arts, Visual and Performing Arts, Movies, Visual Arts, Food, Neighborhood, Community, Religion and Spirituality, Museums and Attractions, Outdoors Recreation, Family and Kids, Galleries and Exhibits, Professional, Fundraising and Charity, Commercial and Sales, Education, Sports, Organizations and Meetups, Business and Networking, Politics, Media and

Literary, Health and Wellness, Life Style, Conferences and Tradeshows, On Campus, Science, Technology, Animals, Comedy, Other and Miscellaneous.

```
eventId : http://data.linkedevents.org/event/143561b1-08ab-4970-9873-
    f80d232ccc07
eventTitle : Lights
publisher : http://www.last.fm
date : 2012-02-24 19:00:00
location : Scala
category : Musical Concert
agent : Nightbox
eventDescription : The event was moved to Scala!
```

For the Eventspotter implementation, we used a modified version of the EventMedia Dataset. We found, upon investigation, that it was necessary to merge multiple records pertaining to the same eventID. The reason why we chose to address this issue was because, when we used the Eventmedia dataset as is, redundant event identifications were created. It is imperitive to note that though these recoreds share the same ID, they differ from time to time in category or agent attributes. In order to tackle this problem, we created event objects for each record, compared their event IDs and merged those objects that had matching eventIDs. Thus during the merge operation, multiple records were transformed into a single record that contained a list of agent strings and list of category strings rather than a single agent string and a single category string. Finally we created a comma separated value (.csv) file that was exported into mySQL. By performing such an operation, the number of records in the new dataset reduced to 34195.

3.1.2 EventTitles

EventTitles.list is a text file which contains, as the name suggests, titles of all events listed in the EventMedia database, mapped to their corresponding eventId. This file is loaded onto memory at the beggining of the execution.

As shown in the figure, the EventSpotter spots candidate entities by matching input text with the eventtitles file and then validates these spots using the agent and location fields from the eventMedia dataset.

3.2 Preprocessing

We first follow a method of text tokenization. Our tokenizer divides the input text as well as a list of event title names into a number of distinct tokens. The tokenizer ensures that there is consistency when it is required to match any 2 strings between the two. Second, the entire input text is also divided into a a set of sentences, a process which we call ‘sentensizeâ. The objective of this process is to ensure the availability of distinct sentences for the later stage of Disambiguation.

3.3 Candidate Selection

For every input text token that was derived from the previous tokenization stage, our system checks for a case-insensitive match with tokens from the list of event title names. When a match is found, the string of tokens are added to a list of matches. Further, only if each match starts with capital letter and first character is not a digit, it is selected as a candidate.

3.4 Disambiguation

In this stage we ascertain the validity of each entry in the candidate list with the help of meta data stored in the eventmedia dataset. Here we make an important hypothesis: It is prudent, for a knowledge based approach to event detection, to identify only those events which are reasonably described in the input text. We consider an event as reasonably described if and only if there is a mention of : *At least one of the agents(performing artists) OR *The location(event venue) in the input text. In other words, this means that many events may go undetected in cases where the agent or the location string is not present in the input text. Though this would affect the recall of the system, we felt that it would be a trade-off worth making. This is primarily because, a large number of events are titled with commonly used words in a language. This means that any strategy that adopts an approach that is less restrictive would end up with a many false positive results. As mentioned earlier, we recognize that basing our approach solely on the first hypothesis, will not allow us to achieve a near-exhaustive detection of event entities in the text. Hence, we intend to deal with these false negatives by employing a linguistic approach in the second pass to detect those events which are not reasonably described in the input text.

For each candidate event, we query the sql database manifestation of the EventMedia dataset to get the names of agents who had performed at that event. We then try to spot these agent tokens in the input text using our naive spotter that we described in the candidate selection stage. If at least one agent is found in the input text then the candidate spot is considered as a valid event and added to the list of detected events. Also for each validated candidate spot we compute a cosine similarity measure between the surroundings of the spot and the event description in the sql database. The event description contains other key information such as location, sentimental expressions such as ¡Add examples¿ , entry fee, domain specific terms such as, ¡Add examples¿, some historical background about the artists, events and so on. These are aspects which are unique to each event and there is a mention of some or all of these attributes in each event description. Thus, it makes logical sense to compare the surroundings of a spot with the corresponding event description. This way we manage to

match various miscellaneous attributes of an event which can, generally not be categorized into fields such as agents, location etc. We empirically defined the surroundings to be the current, previous and next sentences from the spot.This cosine similarity measure is considered as the confidence score for each valid spot. The disambiguation stage yields a list of detected event titles and their respective confidence scores.

3.5 Post processing

In this section we speak about the post processing that was done in order to present the results in a simple and useful manner. Each entry in the detected events list was processed and presented in a json format as shown below.

```
{
                    label: "Pocktoberfest",
                    startChar: 138,
                    endChar: 151,
                    type: "Musical Concert",
                    agent(s): "club smith"
                    uri: "http://data.linkedevents.org/event/86aac5b8
                        -4f0f-41bb-96e9-4377865dc5dd",
                    confidence: 0.82796,
},
```

The uri field points to a web page corresponding to the event on the linkedevents.org website. This page is a web manifestation of the information in the eventmedia dataset. We decided to expose the confidence value to give the users the freedom to allocate any threshold of their choice. Along with the json representation we also annotate the event titles in-place in the input text. To maintain uniformity we follow the same scheme of annotation as defined during the Golden data set creation. For example, ¡add example¿

3.6 Application Pseudocode

The Eventspotter application has been implemented in Java. We show here the assimilated pseudocode for the main logic of the Eventspotter implementation.

```
Input : unstructured text
Output : disambiguated musical events

# When there is input text
tokens = tokenize(input)

sentences = sentensize(input)

candidates = doSpotting(input, tokens, entries)
# entries is a list of tokens obtained by tokenizing the list of
    event titles file
# candidates are case insensitive matches of tokens with entries
```

```
for each candidate in candidates
  if candidate doesnâ t start with capital letter and first char is
      not a digit
          then continue
  end if

    List<String> agents = getAgents(eventId)
  Set<FeatureStructure> foundAgentToks = doSpotting(input, tokens,
      agents)

  Surround = getSurrounding(sentences,candidate);
  #If candidate is in the first line of input text, current and next
      line is surrounding.
  #If candidate is in the middle of input text, previous, current and
       next line is surrounding.
  #If candidate is in the last line of input text, current and
      previous line is surrounding.

  confidence = cosine similarity.run(Surround,event description)

    if at least one agent is found in input text:
    then confirm candidate as an event and add to FeatureStructure.
  endif

endfor
```

4 Evaluation

We tested the Eventspotter against Stanford Conditional Random Field[3] and Opencalais[4]. These tests were of a relatively small scale due to the need to manually create the the golden dataset. We felt it would be interesting to understand the performance of the flexible knowledge based approach of the Eventspotter against the purely grammar-based approaches of Stanford and Opencalais. Before we present the results of our experiments, it would be appropriate to understand the creation of the golden dataset.

4.1 Golden Dataset Creation

We used 60 event descriptions for the creation of the golden dataset. 30 documents were known to include event titles as well as artist names while the remaining were chosen at random. We performed cross validation with factor k =10. We followed a set of rules in order to ensure consistent annotation for the golden dataset creation. For the purpose of annotation we defined a musical event title as : a "proper noun that refers to a musical concert or festival". In all cases, an agreement between 4 human taggers was necessary. Any disagreement was resolved after deliberation and discussion. A candidate spot, was tagged with <EVENT >and </EVENT>. For instance, if 'scala' is an event in the phrase 'scala:)', we annotate it as '<EVENT>scala </EVENT>:)'. For experimental purposes, we followed 2 types of annotation : synthetic annotation and manual annotation . In the case of synthetic annotation, we carried out a straightforward string comparison between the input text and all event titles in the Eventmedia dataset. If there was a match, the spot was annotated as an event. There were several reasons for doing so. First, we were able to initially establish a base for assessing the performance of the Eventspotter. Second, the synthetic annotation was made available for training the Stanford CRF. We will see during the presentation of the results that the Stanford CRF performed quite differently with synthetically annotated training data when compared to manually annotated training data. Third, we realised by observation, that

[3]http://nlp.stanford.edu/software/CRF-NER.shtml
[4]http://www.opencalais.com/

some aspect of the synthetic annotation approach could be applied during the manual annotation process.

As we performed manual annotation, we realised that there was an inherent ambiguity in event titles. We found it extremely difficult, at times, to distinguish between event titles and artist names. Mainly because events were often titled as a mash up of the artist names, venue and sometimes the date of occurrence. Due to this ambiguity many event titles were left untagged despite strict adherence to the annotation rules. Thus, it madesense to not rely solely on the annotators‘ knowledge, but infact, partially adopt the synthetic annotation approach in order to generate a more robust golden dataset. The manual annotation was broken down into two passes.In the first pass, an unbiased, rule based manual annotation was performed. In the second pass, the human annotators were given the list of event titles that were being described in the documents. With this posteriori knowledge the human annotators were able to annotate event occurrences which would have otherwise gone untagged. As human annotators, we leveraged the context of the spot to perform word sense disambiguation. We refrained from annotating those spots where in the event title string was not used to directly refer to the event itself. For instance in the phrase “Carrie Underwood announces her North American tour Blown Away”, ‘Carrie Underwood’ referred to the artist and hence was not annotated. But in another instance, “Carrie Underwood comes to town this monday.” since the annotator knew that Carrie Underwood was the event title and since this string was being used to refer to an event in this context, the spot was annotated as an event. We observed that this hybrid approach towards manual annotation was effective in improving the golden dataset and this was corroborated by the improvement in performance results as presented in the following section.

4.2 Results

The Eventspotter performed better than the Stanford CRF in terms of recall and F-measure. It obtained a score of 89.26% for recall and 72.73% for F-measure, over the scores of 4.27% and 8% respectively of the Stanford CRF trained with synthetic annotation and 54.70% and 60.95% respectively when trained with manual annotation. To highlight once again, these set of tests on the synthetically annotated golden dataset were carried out merely to give ourselves an idea about the Eventspotter’s performance. Opencalais did not identify any musical event entities at all. Opencalais detects events from various categories which does not include musical festival or concert. Since it does come close with the category ‘musical bands’ we thought it would be interesting to note how many events it would detect.

The second set of tests run were run on the manually annotated golden dataset wwe did notice a slight increase in hich contained 115 valid event

Table 1: Tests Performed with Synthetically Annotated Golden Dataset with 122 valid event annotations

Approach	Precision	Recall	F-Measure
Eventspotter	61.36%	**89.26%**	**72.73%**
Stanford trained on synthetic data	62.5%	4.27%	8%
Stanford trained on manual data	**68.82 %**	54.70%	60.95%
Opencalais	no events	no events	no events

annotations. Again the Eventspotter's performance outshined that of the Stanford CRF in terms of recall and F-measure. It obtained a score of 68.14% for recall and 43.14% for F-measure, over the scores of 54.87% and 56.88% respectively of the Stanford CRF trained with synthetic annotation and 11.5% 18.71% respectively when trained with manual annotation. Again, Opencalais did not identify any musical event entities at all. We also tested a linear combination of Eventspotter and Stanford CRF trained on synthetically annotated data, with Stanford CRF acting as a gazateer to Eventspotter. Though there wasn't a stark improvement, the recall score did increase slightly to 70.8%.

Table 2: Tests Performed with Manually Annotated Golden Dataset with 115 valid event annotations

Approach	Precision	Recall	F-Measure
Eventspotter	31.56%	**68.14%**	43.14%
Stanford trained on synthetic data	**59.05%**	54.87%	56.88%
Stanford trained on manual data	50%	11.5%	18.71%
Stanford Gazeteer to Eventspotter	31.25%	**70.8%**	43.36%
Opencalais	no events	no events	no events

5 Conclusion and Future work

In this paper, we presented Eventspotter, a tool for detecting musical event entities in unstructured text. We brought to light the various existent ideas that inspired our approach for the Eventspotter. We then detailed the methodologies implemented followed by the evaluation results. We compared the Eventspotter with other open source services and obtained encouraging results that asserted our belief that there is scope for further research in the direction adopted by Eventspotter.

In the future, we plan to incorporate into the Eventspotter logic, Support Vector Machine classifier and machine learning techniques using WAEKA. Parts of speech tagging, domain specific terms, sentimental expressions and results from the Stanford CRF can be used as features to train the SVM classifier. Also, we can maintain a 'white-list' of verbs which are known occur in the proximity of event title in text. This list can be also used as feature to the SVM classifier. We plan to further extend our knowledge base to include other popular data sources such as DBpedia, DBLP, DBworld etc. We would like to explore the idea of generating.We hope that improvements such as these would be a stepping stone for the evolution of a high performance Eventspotter.

Bibliography

[1] D. Gruhl, M. Nagarajan, J. Pieper, and C. Robson, "Context and domain knowledge enhanced entity spotting in informal text."

[2] J. Hassell, B. Aleman-meza, and I. B. Arpinar, "Ontology-driven automatic entity disambiguation in unstructured text," in *In International Semantic Web Conference*, pp. 44–57, 2006.

[3] P. N. Mendes, M. Jakob, A. GarcÃa-silva, and C. Bizer, "Dbpedia spotlight: Shedding light on the web of documents," in *In Proceedings of the 7th International Conference on Semantic Systems (I-Semantics*, 2011.

[4] A. Ritter, S. Clark, and O. Etzioni, "Open domain event extraction from twitter."

[5] T. Steiner, S. van Hooland, and E. Summers, "Mj no more: Using concurrent wikipedia edit spikes with social network plausibility checks for breaking news detection," *CoRR*, vol. abs/1303.4702, 2013.

[6] H. Khrouf, V. Milicic, and R. Troncy, "EventMedia live: Exploring events connections in real-time to enhance content," in *ISWC 2012, Semantic Web Challenge at 11th International Semantic Web Conference, November 11-15, 2012, Boston, USA*, (Boston, UNITED STATES), 11 2012.