Constructing an Alias List for Named Entities during an Event

Anietie Andy
aandy@seas.upenn.edu
University of Pennsylvania

Mark Dredze
mdredze@cs.jhu.edu
Johns Hopkins

Mugizi Rwebangira
rweba@scs.howard.edu
Howard University

Chris Callison-Burch
ccb@cis.upenn.edu
University of Pennsylvania

Abstract

In certain fields, real-time knowledge from events can help in making informed decisions. In order to extract pertinent real-time knowledge related to an event, it is important to identify the named entities and their corresponding aliases related to the event. The problem of identifying aliases of named entities that spike has remained unexplored. In this paper, we introduce an algorithm, *EntitySpike*, that identifies entities that spike in popularity in tweets from a given time period, and constructs an alias list for these spiked entities. *EntitySpike* uses a temporal heuristic to identify named entities with similar context that occur in the same time period (within minutes) during an event. Each entity is encoded as a vector using this temporal heuristic. We show how these entity-vectors can be used to create a named entity alias list. We evaluated our algorithm on a dataset of temporally ordered tweets from a single event, the 2013 Grammy Awards show. We carried out various experiments on tweets that were published in the same time period and show that our algorithm identifies most entity name aliases and outperforms a competitive baseline.

1 Introduction

Twitter captures a large volume of discussions and messages related to events and topics, in real-time. Knowledge has been extracted from these data streams in different domains to gain various insights, for example, election results (Tumasjan et al., 2010), democratic movements (Starbird and Palen, 2012), tracking illnesses over time (Paul and Dredze, 2011; Guo et al., 2013) and making crucial real-time decisions such as earthquake detection (Sakaki et al., 2010).

Twitter is an informal forum that imposes a limit on the number of characters per tweet, hence, the vocabulary used to express tweets are diverse. This results in the prevalence of abbreviated or misspelled words in tweets and aliases used to represent named entities. Entity name variation poses a challenge to determining what or who a name refers to (Andrews et al., 2014); identifying name variations has been shown to help in different domains such as community question answering systems (Andy et al., 2016b,a) and automatic paraphrase acquisition (Shinyama et al., 2002). For example, given the following tweets that occurred in a 5-minute time period during an event:

- #Veep's Julia Louis-Dreyfus wins for Lead Actress in a Comedy Series #Emmys
- At least Selina Meyer is amazing at winning #Emmys. #Veep
- How does Elaine keep winning these awards?! #Emmys
- Is JLD ever NOT going to win for Veep?#Emmys

where the actress *Julia Louis-Dreyfus* is referred to as *Selina Meyers* — from her character in the show, *Veep*, *Elaine* — from her character in the show, *Seinfield*, and *JLD* — the abbreviation of her name. Identifying these entity name variations can be challenging; however, the context in which the entities occurred and temporal information can be used to determine the aliases of named entities. The goal of our research is to construct an alias list for entities that spike in popularity in a given time period during a single event, by using a temporal heuristic to cluster entities with similar context. This paper makes the following contributions:

- We formulate a temporal heuristic to identify entities with similar context that occur in the same time period during an event.

- We develop a novel algorithm, *EntitySpike*, that uses this temporal heuristic to encode entities as vectors and creates an alias list, if one exists, for entities that spike in popularity in a given time period.
- We present detailed experiments demonstrating that using temporal information identifies most named entity aliases.

2 Background and Preliminaries

In tweets collected during an on going event, there is a small window in time in which entities spike in popularity, though they have occurrences during the whole event (Dredze et al., 2016).

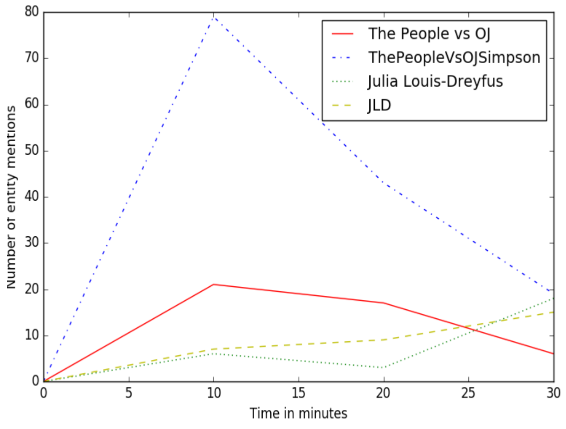

Figure 1: Name variations of entities that spike in popularity at the same time in 10 minute bins during the 2016 Emmy awards show

To investigate the validity of creating an alias list for entities that spike, we use the Twitter streaming API to collect 50,000 timestamped and temporally ordered tweets and re-tweets containing *#emmys*, while the 2016 Emmy Awards show was going on (we intend to make this dataset available to the research community). We selected tweets that occurred in a randomly selected 30 minutes time period and split these tweets into bins of 10 minute time periods i.e. each bin contained tweets that occurred in a 10 minute time interval. Figure 1 shows the entities that spiked the most in this time period and their name variations. We observed that when an entity spikes in popularity in a given time period, some of its aliases spike in popularity as well. Based on these observations, we propose an algorithm, *EntitySpike*, that creates an alias list for entities that spike in a given time period.

Task Definition: Given a sequence of tweets and entity mentions, denoted by $\mathbf{X}$ =($\{\mathtt{e}_1, \mathtt{S}_1\}$,$\{\mathtt{e}_2, \mathtt{S}_2\}$,....$\{\mathtt{e}_n, \mathtt{S}_n\}$), where e_i represents a named entity that spikes in popularity in a given time period e.g. *Julia Louis-Dreyfus*, and S_i represents the set of tweets that make reference to this named entity, e_i, during this specified time period; the task is to create an alias list for each e_i, if one exists.

Candidate Entity Identification: Following previous work in entity linking (Liu et al., 2013; Guo et al., 2013), we define an entity as a Wikipedia title page. An entity mention is a sequence of tokens in a tweet that can potentially link to an entity. The Grammy Awards show is mostly about famous people and so we focus only on entities belonging to the *Person* category. In order to construct a Wikipedia lexicon, we collect 1.5 million English Wikipedia title pages referring to *Person* named entities and extracted the backlinks (incoming links to the Wikipedia title page) from each of these Wikipedia title pages (we intend to make this dataset available to the research community).

Given a set of tweets $\{t_1, t_2, ..., t_n\}$ that occur in a given time period (within minutes) during the Grammy awards show, we extract all *k-grams* of size $\leq k$, from each tweet; we selected $k = 3$. We select a *k-gram* as a candidate entity mention if it is either an exact match of a Wikipedia title page or backlink or if it is contained in a Wikipedia title page or backlink, for example ”Carrie” is contained in ”Carrie Underwood”.

3 Our Algorithm: *EntitySpike*

During an event, such as an award show, a large volume of tweets related to the event are generated by users, per minute. Most of these tweets contain named entities and their aliases. Our algorithm, *EntitySpike* identifies named entities that spike at a certain time period and constructs an alias list for these entities by using a temporal heuristic.

Exploiting Temporal Information about entities: While the award show is going on, artists and celebrities - some of which have nicknames, are showcased walking on the red carpet, performing on stage, presenting awards, or sitting in the audience. On social media platforms, such as twitter, people publish tweets related to these events, in real-time. Most of the generated tweets refer to people or entities in the same context e.g. their outfit, performance, actions etc. Some tweets make reference to named entities by their aliases, some of which look similar to their corresponding named entity e.g. *"Jay Z"* and *"Jay-Z"* and some look different e.g. *"Taylor Swift"* and *"Tswizzle"*. Given tweets that occur at a certain time period (within minutes), the context of most of the tweets will be similar. For example, one may see a burst of tweets published in the same 5-minute time period that say: *"**X** just won a Grammy"* or *"**Y** won her first Grammy"*. This suggests that ***X*** and ***Y*** refer to the same person, regardless of how different the names are. Based on this intuition, we propose the *Temporal Entity Similarity* heuristic:

Temporal Entity Similarity: *In tweets (related to an event), collected sequentially, named entities that occur at the same time period and have a similar context are referring to the same named entity.*

This temporal entity similarity heuristic helps capture the temporal context in which entities are mentioned and clusters entities that occur in a similar context. In figure 2, we present an outline of *EntitySpike* which uses the temporal entity similarity to create an alias list for named entities that spike in popularity in a given time period during an event. To measure the similarity between the named entities, we represent each entity as a vector. To create the vector for each entity, we select all the unique words in the *EntitySpike*, $(\{\mathtt{e_1, S_1}\},\{\mathtt{e_2, S_2}\},...,\{\mathtt{e_n, S_n}\})$ and count the frequency of occurrence of each word in tweets, S_i related to a named entity e_i. Two named entities refer to the same entity if the cosine similarity of their vectors is greater than a threshold.

Algorithm 1 EntitySpike

Given $\mathbf{X}$=$(\{\mathtt{e_1, S_1}\},\{\mathtt{e_2, S_2}\},....,\{\mathtt{e_n, S_n}\})$, were **X** satisfies the temporal entity similarity heuristic
Output Alias list for e_i

```
1: procedure EntitySpike(X)
2:     for each entity $e$ in X do
3:         create temporal vector, $\mathbf{V_e}$
4:     for each $v$ in $\mathbf{V_{e,...,n}}$ do
5:         Cosine Similarity($v$, $\mathbf{V_{e,...,n}}$)
6:         if Cosine Similarity $>$ threshold then
7:             Insert into Aliaslist($\mathbf{V_e}$)
8:     Return Aliaslist($\mathbf{V_e}$)
```

Figure 2: Temporal Entity Similarity Algorithm

4 Experiments

4.1 Data

For our experiments, we evaluated *EntitySpike* on the 2013 Grammy Awards Show dataset described in Dredze et al. (2016). The show lasted for approximately three and a half hours generating a lot of tweets, most of which made reference to artists, celebrities, and famous people - with entries in Wikipedia, by their names and nicknames (Dredze et al., 2016). To create this dataset, Dredze et al. (2016) used the Twitter streaming API to collect 10,736 temporally ordered and unique tweets written in English containing *grammy* (case insensitive, and including *#grammy*) during the event. Although these tweets were temporally ordered, their timestamps were not saved, hence, we split the dataset into equal size temporal bins (11 bins), with each bin containing 976 temporally ordered tweets. We represented each entity as a vector by collecting the frequency of occurrence of each entity in each bin i.e. each entity was represented as a vector of size 11.

4.2 Baseline

Before carrying out experiments for *EntitySpike*, we conducted preliminary experiments to show that entities that frequently co-occur together should have a high cosine similarity. The intuition here is that entities with a cosine similarity above a threshold could be referring to the same entity. Based on this intuition, we represented each entity as a vector by collecting the frequency of its occurrence in each bin. We calculated the cosine similarity between all of the named entities and ranked them. For each entity, we select the 10 highest ranked similar entities. For evaluation of this algorithm, we selected and labelled 40 named entities and their known aliases in our dataset. On these named entities and their aliases, this algorithm had a precision of *68.25%* and a recall of *43.1%*.

This algorithm shows that some entities that frequently occur in the same context at different time periods are referring to the same entity. We use this algorithm as our baseline.

Word2Vec Experiments: Given a named entity, we used *word2vec* as implemented by Mikolov et al. (2013) to find the top 10 most similar entities to the given entity. We observed that *word2vec* found related words and entities to a given entity, however, it found few entity aliases, some of which are misspelled words and abbreviations. Table 1 shows the top 2 related entities found by *word2vec* for some named entities.

Named Entity	Related Entities
Justin Timberlake	Timberlake, Beyonce
Adam Levine	Maroon 5, Rob Thomas
Taylor Swift	Miley Cyrus, Justin Bieber
JayZ	Music Beats, Young Joc

Table 1: Some named entities and related entities identified by *word2vec*

4.3 *EntitySpike* Experiments

As stated in section 4.1, we split our dataset into 11 equal size bins. In each bin, we calculated the frequency of occurrence of each entity and selected the top *u* most frequently occuring entities. We chose $u = 15$ because we wanted to compare a relatively small set of entities. We represented each of these entities using the method described in section 3 and calculated the cosine similarity between these entities. We selected an entity as an alias to another entity if they had a cosine similarity greater than a threshold (0.95). We evaluated *EntitySpike* on the named entities that spiked in each of the 11 bins. We labelled the aliases of these spiked entities - that were found in the dataset. We used the algorithm described in section 4.2 as a baseline. Table 2 shows the precision and recall (with respect to the spiked entities and their aliases) of *EntitySpike* and the baseline.

Algorithm	Precision	Recall
EntitySpike	73%	65%
Baseline	65%	69%

Table 2: Precision and recall of *EntitySpike and baseline*

Table 3 shows aliases of the named entities from table 1, that *EntitySpike* identified. It can be seen that *EntitySpike* found more aliases.

Named Entity	Alias
Justin Timberlake	Justin T, Timberlake
Adam Levine	Adam Levin, Adam
Taylor Swift	Taylor S,Taylor
JayZ	Jay Z, Jay-Z

Table 3: Some named entities and their aliases identified by our method

EntitySpike identified entities that spiked in a given time period and created an alias list for these entities. We conducted experiments that varied this time period i.e. we combined bins, and we observed that some spiked entities persisted across bins and combining two adjoining bins gave optimal results.

EntitySpike also identified related words. For each entity that spiked, we used *EntitySpike* to calculate the cosine similarity between them. Two entities were related if they had a cosine similarity above a threshold (0.89). For each spiked entity, we compared the results from this experiment with *word2vec* and *Entity Spike* found some of the related entities in *Word2vec*. For example, it identified that during the show, *Jayz* and *Beyonce* were related and also, *Chris Brown* and *Rihanna*. During the event, some entities were more related than usual; for example, *Jay Z* and *Justin Timberlake* preformed together during the show and a lot of tweets - referring to both of them were generated at this time period. *EntitySpike* was able to identify that these two entities were related at this particular time period. *EntitySpike* also identified some named entities that did not occur in Wikipedia e.g. ”Kelly C” for ”Kelly Clarkson” and ”Rih” for ”Rihanna”.

5 Conclusion and future work

In conclusion, we proposed an algorithm that creates an alias list for entities that spike. We conducted experiments to show that our algorithm finds most entity aliases by using a temporal heuristic. In the future, we will research using temporal information to detect how relationships between entities change over a period of time.

References

Nicholas Andrews, Jason Eisner, and Mark Dredze. 2014. Robust entity clustering via phylogenetic inference. In *ACL (1)*. pages 775–785.

Anietie Andy, Mugizi Rwebangira, and Satoshi Sekine. 2016a. An entity-based approach to answering recurrent and non-recurrent questions with past answers. *OKBQA 2016* page 39.

Anietie Andy, Satoshi Sekine, Mugizi Rwebangira, and Mark Dredze. 2016b. Name variation in community question answering systems. *WNUT 2016* page 51.

Mark Dredze, Nicholas Andrews, and Jay DeYoung. 2016. Twitter at the grammys: A social media corpus for entity linking and disambiguation. In *Proceedings of the 4th Workshop on Natural Language Processing and Social Media*. pages 20–25.

Stephen Guo, Ming-Wei Chang, and Emre Kiciman. 2013. To link or not to link? a study on end-to-end tweet entity linking. In *HLT-NAACL*. pages 1020–1030.

Xiaohua Liu, Yitong Li, Haocheng Wu, Ming Zhou, Furu Wei, and Yi Lu. 2013. Entity linking for tweets. In *ACL (1)*. pages 1304–1311.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* .

Michael J Paul and Mark Dredze. 2011. You are what you tweet: Analyzing twitter for public health. *Icwsm* 20:265–272.

Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. 2010. Earthquake shakes twitter users: real-time event detection by social sensors. In *Proceedings of the 19th international conference on World wide web*. ACM, pages 851–860.

Yusuke Shinyama, Satoshi Sekine, and Kiyoshi Sudo. 2002. Automatic paraphrase acquisition from news articles. In *Proceedings of the second international conference on Human Language Technology Research*. Morgan Kaufmann Publishers Inc., pages 313–318.

Kate Starbird and Leysia Palen. 2012. (how) will the revolution be retweeted?: information diffusion and the 2011 egyptian uprising. In *Proceedings of the acm 2012 conference on computer supported cooperative work*. ACM, pages 7–16.

Andranik Tumasjan, Timm Oliver Sprenger, Philipp G Sandner, and Isabell M Welpe. 2010. Predicting elections with twitter: What 140 characters reveal about political sentiment. *ICWSM* 10(1):178–185.