

Pattern Matching

CPE 393: Text Analytics

Dr. Sansiri Tarnpradab

*Department of Computer Engineering
King Mongkut's University of Technology Thonburi*

What is Pattern Matching?

- A mechanism for checking a value against a pattern
- **Input:** sequence of tokens

Why Pattern Matching?

1607.00976v2 [cs.CL] 5 Jul 2016

Modelling Context with User Embeddings for Sarcasm Detection in Social Media

Silvio Amir¹ Byron C. Wallace¹ Hao Lyu¹ Paula Carvalho² Mário J. Silva²

¹INESC-ID Lisboa, Instituto Superior Técnico, Universidade de Lisboa

²School, University of Texas at Austin

samir@inesc-id.pt byron.wallace@utexas.edu xalh8083@gmail.com

pcc@inesc-id.pt mjs@inesc-id.pt

Abstract

We introduce a deep neural network for automated sarcasm detection. Recent work has emphasized the need for models to capitalize on *contextual* features, beyond lexical and syntactic cues present in utterances. For example, different speakers will tend to employ sarcasm regarding different subjects and, thus, sarcasm detection models ought to encode such speaker information. Current methods have achieved this by way of laborious feature engineering. By contrast, we propose to automatically learn and then exploit *user embeddings*, to be used in concert with lexical signals to recognize sarcasm. Our approach does not require elaborate feature engineering (and concomitant data scraping); fitting user embeddings requires only the text from their previous posts. The experimental results show that our model outperforms a state-of-the-art approach leveraging an extensive set of carefully crafted features.



Figure 1: An illustrative tweet.

Enquiring Minds: Early Detection of Rumors in Social Media from Enquiry Posts

Zhe Zhao
Department of EECS
University of Michigan
zhezha@umich.edu

Paul Resnick
School of Information
University of Michigan
presnick@umich.edu

Qiaozhu Mei
School of Information
University of Michigan
qmei@umich.edu

ABSTRACT

Many previous techniques identify trending topics in social media, even topics that are not pre-defined. We present a technique to identify trending rumors, which we define as topics that include disputed factual claims. Putting aside any attempt to assess whether the rumors are true or false, it is valuable to identify trending rumors as early as possible.

It is extremely difficult to accurately classify whether every individual post is or is not making a disputed factual claim. We are able to identify trending rumors by recasting the problem as finding entire clusters of posts whose topic is a disputed factual claim.

The key insight is that when there is a rumor, even though most posts do not raise questions about it, there may be a few that do. If we can find signature text phrases that are used by a few people to express skepticism about factual claims and are rarely used to express anything else, we can use those as detectors for rumor clusters. Indeed, we have found a few phrases that seem to be used exactly that way, including: "Is this true?", "Really?", and "What?". Relatively few posts related to any particular rumor use any of these enquiry phrases, but lots of rumor diffusion processes have some posts that do and have them quite early in the diffusion.

We have developed a technique based on searching for the enquiry phrases, clustering similar posts together, and then collecting related posts that do not contain these simple phrases. We then rank the clusters by their likelihood of really containing a disputed factual claim. The detector, which searches for the very rare but very informative phrases, combined with clustering and a classifier on the clusters, yields surprisingly good performance. On a typical day of Twitter, about a third of the top 50 clusters were judged to be rumors, a high enough precision that human analysts might be willing to sift through them.

Categories and Subject Descriptors

Keywords

Rumor Detection; Enquiry Tweets; Social Media

1. INTRODUCTION

On April 15th of 2013, two explosions at the Boston Marathon finish line shocked the entire United States. The event dominated news channels for the next several days, and there were millions of tweets about it. Many of the tweets contained rumors and misinformation, including fake stories, hoaxes, and conspiracy theories.

Within a couple of days, multiple pieces of misinformation that went viral on social media were identified by professional analysts and debunked by the mainstream media.¹ These reports typically appeared several hours to a few days after the rumor became popular and only the most widely spread rumors attracted the attention of the mainstream media.

Beyond the mainstream media, rumor debunking Websites such as Snopes.com and Politifact.org check the credibility of controversial statements.² Such Websites heavily rely on social media observers to nominate potential rumors which are then fact-checked by analysts employed by the sites. They are able to check rumors that are somewhat less popular than those covered by mainstream media, but still have limited coverage and even longer delays.

One week after the Boston bombing, the official Twitter account of the Associated Press (AP) was hacked. The hacked account sent out a tweet about two explosions in the White House and the President being injured. Even though the account was quickly suspended, this rumor spread to millions of users. In such a special context, the rumor raised an immediate panic, which resulted in a dramatic, though brief, crash of the stock market [10].

The broad success of online social media has created fertile soil for the emergence and fast spread of rumors. According to a report of the development of new media in China, rumors were detected in

Pattern Matching: How

- **Regular Expressions**
 - A method for describing the character patterns of interest.
 - Powerful and flexible
 - Purpose:
 - To *find* a substring
 - To *replace* a substring
 - To *segment* a string
- **Tools**
 - Online: <https://regex101.com/>
 - Python: `re` library
 - Java: `java.util.regex`
 - R: `stringr`

Regular Expression

RegEx Syntax

- **What are they** – the common ones
- **How to write** RegEx to find/match a pattern that we are looking for

First thing first...

Every character matches itself;

So does every word.

Any match

(.)

- A dot
- A period
- Matches any character

S.t

- Sat, Set, Sit, Sbt, Sct, Sdt
- Uppercased S, lowercased t



c.t

- cat, cut, cute, caterpillar
- Lowercased c and t

Where it *begins*

‘^’

- A caret symbol
- A hat
- A circumflex
- Indicate the start of a string

^a...

- 4-letter word starting with ‘a’
- **Perfectly Matched:** aged, amid, able, away, army, area, also, atoms
- **Partially matched:** amidst, atomic
- **Unmatched:** capable, Aged, Amid

Where it *ends*

‘\$’

- A dollar sign symbol
- Indicate where the string ends

d\$

Agedu, agedu, landu, Acceptedu

land\$

~~Aged~~, ~~aged~~, land, ~~Accepted~~, Finland,
Disneyland, Poland, LaLaland

ed\$

Acceptedu, acceptedu, wickedu, masteredu,
~~educate~~, educatedu

Or

‘|’

- Pipe
- Matches either one

this|that

- **Matched**: this, that
- **Unmatched**: thisthis, thisthat

Character Set


‘[]’

- Matches any single character in the bracket

[abc]

- **Matched:** a, b, c
- **Unmatched:** ab, ac, bc, cb, abc

[abc]at

- **Perfectly Matched:** bat, cat 
- **Partially Matched:** caterpillar
- **Unmatched:** act, bacteria

[^abc]at

- A negated set – match any single character that is not in the set
- **Matched:** mat, matter, spatula



Range

(_)

- A dash
- A hyphen
- To specify range:
 - [A-Z] → within a range A to Z
 - [a-z] → within a range a to z
 - [A-Za-z] → both lower&upper cases
 - [0-9] → within a range zero to nine

[A-Z]at

- **Matched:** Cat, Cats
- **Unmatched:** cat, at, you_name_it

[A-Za-z]at[0-5]

- **Matched:** Cat1, cat2
- **Unmatched:** 1Cat, 2cat, you_name_1

Number of Preceding Tokens



- Matches the previous token between one and unlimited times
- At least one

a+

- **Matched:** a, aa, aaa, aaaaaaa
- **Unmatched:** '', b, sldkfjsldkfj

cat+

- **Matched:** cat, cattttt, catratdog
- **Unmatched:** cart, you_name_it

Number of Preceding Tokens

(*)

- Matches the previous token between zero and unlimited times
- Arbitrary

a*

- **Matched:** '', a, aa, aaa, aaaaaaa
- **Unmatched:** b, sldkfjsldkfj

cat*

- **Matched:** cat, catttttt, catratdog, cart
- **Unmatched:** you_name_it



Number of Preceding Tokens

‘?’



- Matches the previous token between zero and one times

a?

- **Matched:** '', a
- **Unmatched:** aa, aaa, aaaaaaa, b, sldkfjsldkfj

cat?

- **Matched:** cat, cattttt, catratdog, cart
- **Unmatched:** you_name_it

Special Characters

{_}

{_,_}

- Specify number of preceding tokens
- {m} → exactly m tokens
- {m,n} → From m to n tokens

a{1,3}

- **Matched:** a, aa, aaa
- **Unmatched:** aaaaaaa, b, sldkfjsldkfj

cat{1,3}

- **Matched:** cat, cattttt, catratdog
- **Unmatched:** cart, you_name_it



Group of Tokens

'()'



- Match all tokens in the parenthesis

(cat)+

- **Matched:** cat, catcat, catttttt, catratdog
- **Unmatched:** '', cart, you_name_it

(cat)*

- **Matched:** '', cat, catcat, catttt, catratdog
- **Unmatched:** cart, you_name_it

(cat)?

- **Matched:** '', cat, catttttt, catratdog
- **Unmatched:** cart, you_name_it



RegEx Symbols

Symbol	Function
\b	Word boundary (zero width)
\d	Any decimal digit (equivalent to [0-9])
\D	Any non-digit character (equivalent to [^0-9])
\s	Any whitespace character (equivalent to [\t\n\r\f\v])
\S	Any non-whitespace character (equivalent to [^ \t\n\r\f\v])
\w	Any alphanumeric character (equivalent to [a-zA-Z0-9_])
\W	Any non-alphanumeric character (equivalent to [^a-zA-Z0-9_])
\t	The tab character
\n	The newline character

Simple Challenge

RegEx to match

- Year (e.g. 2023)

Another Challenge

RegEx to match

- words ending with 'ed' or 'ing'

Challenging one...

RegEx to match

- Construct a regex that captures word start and end with vowel (a, e, i, o, u)

Homework

- Propose 3 sensible RegExes (should be challenging and meaningful, not too plain)
- Show your work in the Jupyter notebook
- For each RegEx, briefly describe:
 - Why you want to discover those patterns?
 - Why are they interesting to you?
 - Does those RegExes work as you expected?
- Show your results

Q&A

All cat pictures are from:

<https://www.freeiconspng.com/images/cat-png>

Cheat sheet

Download [link](#)

