CS246: Twitter Purification

Wen Shi, Zijun Xue, Jennifer Zhang

June 2014

Abstract

Twitter is one of the most prolific and widely available sources of social media data; 1 billion tweets are produced every 5 days.[1] However, it often behaves badly as a dataset due to the high levels of noise, and nearly any project that utilizes tweets as a dataset must prepend their analysis with some data normalization. The goal of this project is to purify tweets of misspellings, abbreviations, etc. to potentially convert it into a more convenient source of data for other research purposes. The problem of common unintentional misspellings is relatively well-solved, but tweets are particularly malformed for a host of other reasons.

Two analysis passes of a potentially malformed tweets are used: once for single word correction, and the second for contextual analysis. For single word correction, we use a combination of common abbreviations, word frequency, dictionary presence, and distance between a potential correction candidate and the original word. Distance is defined in terms of edit distance, phonetic class, and overall letter similarity. Contextual analysis uses bigram frequency to find the best candidate in cases where single word correction still left ambiguity.

Implementation

Dataset

Approximately 800,000 tweets consisting of about 120,000 distinct words were harvested Twitter's sample API[2]. Word tokens were defined as any whitespace delimited string that begins with an alphabet letter and otherwise may include alphanumeric characters, apostrophes, or hyphens. We excluded some Twitter specific tokens, including usernames (strings that begin with @), hashtags (strings that begin with #), emoticons, and internet urls (*http(s)://...*). All strings were converted to lowercase for analysis. Final corrected output replaced any usernames with]*username*], and a hardcoded list of laughter tokens were converted to [*laughter*].[3]

Dictionary

In a standard spellchecker, a word would be searched against an existing dictionary, and skipped over if it exists and corrected if it doesn't. This is, unfortunately, not an acceptable method of attack for correcting tweets. The major issue with this approach is that it assumes a perfect dictionary; given the amount of slang, proper nouns, etc. in tweets, this is not only exceptionally difficult, but constitutes a moving target as new slang and names are propagated daily.

In the case where a given word already exists in the dictionary, it could easily be a word misuse; aside from common grammatical mistakes such as confounding *your* or *you're*, it is common to intentionally misspell *then* as *den*. Since *den* is a legitimate word in the English language, a normal spellchecker would fail to attempt to correct it.

In the case where a word is encountered that does not exist in a dictionary, it is often more appropriate to not apply a correction. Slang and proper nouns abound in Twitter; while the line between a legitimate word versus a misspelling can be blurry, e.g. *want to* vs *wanna*, but a great deal of slang have no 'correct' equivalent in a dictionary, e.g. Twitter specific words such as *retweet* as well as many more vulgar examples.

Nonetheless, a high quality dictionary was desirable for this project. Word candidates that exist in the dictionary can therefore be weighted more heavily, and correctly spelled obscure words can avoid being overwhelmed by exceptionally common words that are closely phonetically related or via edit distance (e.g. *goat* being overwhelmed by the frequency of *got*. Initially, a union of UNIX's dictionary, Google Translate's list of most frequently used words[4], and the most common words from TV and movie scripts[5] was used. However, this failed to produce an acceptable dictionary, partially due to the overinclusion of slang and underinclusion of conjugated verbs or plurals. Inevitably, we settled upon using Ispell's english dictionary, which consisted of 143,006 words.[6]

In the context of Twitter purification, if the given word exists in the dictionary, it is guaranteed a position in the final list of correction candidates, with the same weight as that of the frontrunner candidate (if different). Furthermore, any candidates that exist in the dictionary are weighted more heavily by +20%.

[1]Twitter Statistics
[2]GET statuses/sample
[3]lmao, lmfao, lolz, lol, rofl, lls, haha
[4]google-10000-english
[5]Wiktionary Frequency Lists
[6]Dictionaries for International Ispell

Single word correction

Abbreviations: single word

The most common occurrences in the Twitter dataset were crosschecked against the dictionary, and mismatches were examined. Words that are abbreviations for single words[7] such that their distance was judged to be too far for an automated system to determine were listed into a separate reference file[8]. This included instances where an abbreviation potentially had multiple interpretations (e.g. *ur* $\rightarrow$ [*your, you're*]).

When the purifier begins analysis of a word, it is crosschecked against this manually maintained list and makes replacements in the correction candidate list as indicated.

Squeeze

TODO: Zijun

Edit distance

TODO: Zijun

Phonetic candidates: Soundex

TODO: Shi Wen

Phonetic candidates: Metaphone

TODO: Shi Wen

Letter similarity: Viterbi

Several hundred correction candidates may be found for a word, based on searching within a particular edit distance or the same Soundex/Metaphone class. To trim this list, we first make some intuitive assumptions about the manner in which Twitter users typically misspell their words.

- Abbreviation: Twitter words are usually shorter than their correct counterparts
- Letter similarity: the same important letters are usually present in the Twitter word as the correct counterpart. There are a few phonetic exceptions to this, e.g. substituting *d* for *th*, as in *dere* vs *there*.
- Transposition of remaining significant letters is rare, which makes sense if other letters have already been omitted for brevity.

With these observations in mind, we set out to find a reasonable scoring algorithm to measure the similarity between a Twitter word versus its correction candidate.

The Viterbi algorithm is a dynamic programming algorithm typically used in hidden Markov models, such that it finds the optimal path of hidden states given a set of observations. In implementation, an $n \times m$ matrix V is created, such that n is the number of observations and m is the number of hidden states. Each entry of the matrix is a product of the probability of transitioning from the prior state into that state multiplied by the Viterbi score of the prior state; the maximum of these possibilities is populated into the matrix. Mathematically, where t is the timestep, i is the hidden state, and j is the prior hidden state:

$$V_{ti} = \max_j p(X_t = s_i | X_{t-1} = s_j) V_{t-1,j}$$

When the exact path of hidden states needs to be returned, another matrix with identical dimensions is needed to keep track of which prior state was chosen for each entry.

In the context of Twitter purification, the Viterbi matrix is constructed such that n is length of the observed tweeted word, and m is the length of the correction candidate. The transition probability to a specific entry is interpreted as the score to match two letters, or to skip a letter in either direction. In the interest of computational simplicity, addition is used as the logarithmic equivalent to multiplication; this is common practice when probability values are involved for better computational handling of small numbers. Thus, where w is the Twitter word and c is the correction candidate:

$$V_{j,i} = \max \begin{cases} \text{match score between } (w_i, c_j) + V_{j-1,i-1} \\ \text{gap penalty of skipping over } w_i + V_{j,i-1} \\ \text{gap penalty of skipping over } c_j + V_{j-1,i} \end{cases}$$

The value of $V_{m,n}$ is interpreted as the best match score between w and c.

Transition costs are set in a 27×27 matrix that indicate the positive score of a match or negative penalty (see Appendix A-2). Since we only have interest in relative scores rather than actual probabilistic values, the individual numbers do not need to be normalized in any sense. The horizontal and vertical gap penalties are $(-1, -0.1)$ respectively, to reflect the fact that Twitter words are more likely to be abbreviated. Both the scoring matrix and appropriate gap penalties were roughly approximated based on observations about misspellings that were encountered, and have a great deal of room for improvement in a more analytic manner.

An example of calculating the letter similarity of *den* versus correction candidate *then*, excluding irrelevant entries:

$$\begin{array}{c|cccc}
 & & d & e & n \\
\hline
 & 0 & \leftarrow -1 & \leftarrow -2 & \leftarrow -3 \\
t & \uparrow -0.1 & \nwarrow 4 & & \\
h & \uparrow -0.2 & \uparrow 3.9 & & \\
e & \uparrow -0.3 & & \nwarrow 8.9 & \\
n & \uparrow -0.4 & & & \nwarrow 13.9
\end{array}$$

The final similarity score is 13.9/15.0, where 15.0 is the similarity of (*den, den*). Each correction candidate is weighted by their similarity scoring, and any with a similarity score less than 0.7 are thrown out.

[7] Common abbreviations for phrases were stored separately for substitution later

[8] see Appendix A-1

This algorithm does not behave as well in cases where the Twitter word is longer than the original. In general, this is rare, but a common exception is replacing the IPA phoneme α[9] with *aw*, e.g. writing *dawg* instead of *dog*. An idealized version of this algorithm might use phonemes instead of specific letters, but this would require an additional conversion step from a given English word to potential pronunciations. While some phonetic dictionaries for the English language exist[10], there would need to be an algorithm that interprets the pronunciation of the obscure, misspellings, or slang among the Twitter input.

Word frequency scaling

Using Bayes' rule, where w is the tweeted word and c is the candidate correction:

$$\Pr(c|w) \propto \Pr(c)\Pr(w|c)$$

$\Pr(w|c)$ represents the weights associated with candidates thus far – it includes some accounting for edit distance and/or phonetic distance as well as letter similarity to the original. $\Pr(c)$ is now applied using the word frequency of c in the Twitter dataset. A hash table of all words and their respective frequencies was created ahead of time so lookups were $O(1)$.

Originally, the weight was multiplied by the raw word frequency, but this quickly proved to produce undesirable results; it became impossible to prevent common words from overwhelming less common words, e.g. *got* has a word frequency of 13807, and *goat* has a word frequency of 87, approximately 2 orders of magnitudes apart. This particular example isn't troublesome since *goat* exists in the dictionary and would be restored to the candidate list, but this problem is particularly serious for any words that the dictionary does not cover. Therefore, the weight was multiplied by ln(word frequency) to reduce its impact.

As a final step, the list of correction candidates are trimmed down to the most promising ones. Suppose that s_i is the highest score out of the correction candidates. Any correction candidates with a score $s_j > 70\% \times s_i$ are kept.

Bigram correction

TODO: add some subsections here?

Substitutions

During the analysis for common abbreviations described above, any common abbreviations that map to phrases were stored in a separate reference file. Note that unlike single word abbreviations, this list was limited to unambiguous abbreviations, and only map to a single possibility (see Appendix A-3). After single word and bigram correction, both laughter tokens and any such abbreviations were substituted with these expansions.

[9]Wiktionary: α
[10]The CMU Pronouncing Dictionary

Discussion and Evaluation

Single word results

368 of the most frequent Twitter words that did not appear in the dictionary were manually mapped to their correct counterparts. In some cases, there were multiple possible corrections; let us refer to this set of corrections as L_1. The single word correction algorithm as detailed above was applied to each word in turn, and the resultant list of candidates L_2 was compared with L_1. If $L_1 \cap L_2 = \varnothing$, then it is counted as an error. The Twitter purification algorithm above produced 76/368 errors. TODO: Zijun or Shiwen to add some data about comparison with existing spell checker

Bigram results

Future work

TODO: listing any more aspects where it didn't work well

Unconventional word tokenization

The spell checking mechanism as described here works primarily for single words that are misspelled or malformed into another single word token. It does not handle cases where multiple words have been merged together in the misspelling, e.g. *tryna*, *whaddya*. However, we noted that the ISpell dictionary does not flag other more common instances of the same concept as a misspelling, e.g. *wanna, gonna*. Thus, it could potentially be argued that since these portmanteaus have a higher tendency to cross into common vernacular, correcting them into their more formally correct versions is not high priority.

More rarely, tweets may contain multiple word tokens meant to be read as a single word, e.g. with spaces between every character, as in *p r e t t y*. Correcting spacing in the most general manner might use a dynamic programming algorithm that removes all whitespace, then adds the spacing back in based on creating known words.[11] However, this solution is $O(n^2)$ complexity; furthermore, given the raw tweet data, there is no guarantee that there may not be entirely new words in the tweet that complicate this process. Alternatively, assuming that extra spacing always follows the same pattern where single characters are separated into individual word tokens might be used to correct this, e.g. if 3+ consecutive single letter word tokens are found, try merging them into a single word.

Proper nouns and obscure slang

Both proper nouns and obscure slang are often badly handled due to their rarity and lack of presence in the dictionary. For example, *Cersei* is the name of a television show character which gets corrected to either *course* or *curse*. It might be possible to allow leniency for proper nouns and acronyms if we took capitalization into consideration. A dictionary of

[11]Dynamic Programming — Set 32 (Word Break Problem)

proper nouns could be maintained that is based on the frequency of the word occurring with or without capitalization; entirely novel words that are capitalized would be exempt from correction. The size of the Twitter dataset might need to be much larger.

However, capitalization metadata is not useful for determining the legitimacy of obscure slang. The word *thot* is meant to be an acronym for 'that ho over there,' although its usage has degraded to a general derogatory term for women. The Twitter purifier corrects *thot* to *thought* or *that*; since the original word is so closely phonetically related to both of these correction candidates and occurs far less commonly, it is highly difficult to avoid a correction without adding the word to the dictionary. Tweets are a moving target in terms of an applicable dictionary; while Urban Dictionary[12] might provide some assistance here, it is another crowdsourced database that is difficult to navigate successfully using an automated system.

Abbreviations

It is particularly difficult to account for phrase abbreviations that mean differing things in different contexts. Examples of this can stretch from *af* being any of [*at, a fuck, as fuck*] to *kd* being *Kevin Durant* during a sporting event, or *kill/death* in relation to a gaming event.

Related work

Aspell's algorithm uses a combination of DoubleMetaphone and weighted edit distance to find correction candidates.[13] Search engines such as Google use a more similar method to what was outlined here, using a combination of probabilistic data and edit distance.

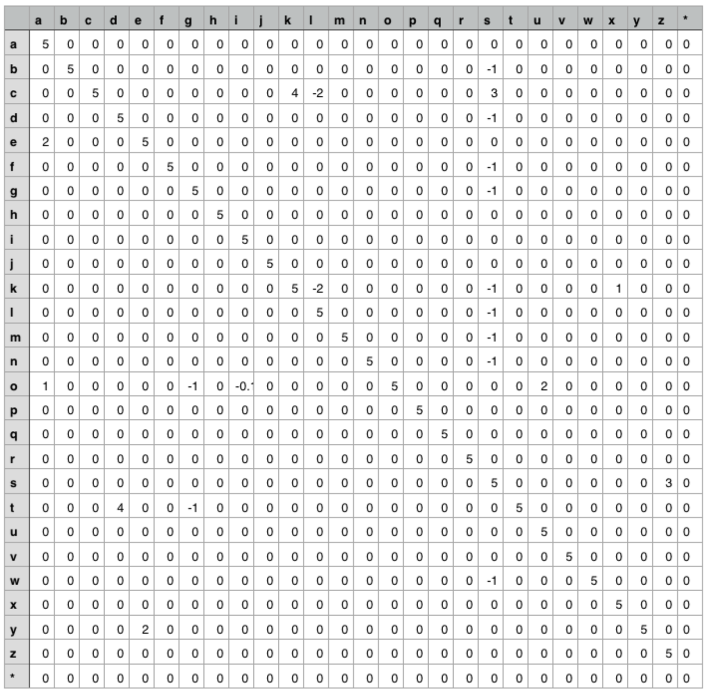
Appendix

A-1 Abbreviations: single word

`2`	`to`
`2`	`two`
`2`	`too`
`fav`	`favorite`
`xd`	`XD`
`tl`	`timeline`
`ig`	`instagram`
`fb`	`facebook`
`u`	`you`
`r`	`are`
`c`	`see`
`y`	`why`
`ur`	`your`
`ur`	`you're`
`w`	`with`
`bf`	`boyfriend`
`gf`	`girlfriend`
`bfs`	`boyfriends`
`gfs`	`girlfriends`
`gr8`	`great`
`bc`	`because`
`mfs`	`motherfuckers`
`mf`	`motherfucker`
`rt`	`retweet`
`hw`	`hw`
`hw`	`homework`
`rp`	`roleplay`
`da`	`the`
`da`	`da`

[12]Urban Dictionary
[13]8. How Aspell Works

A-2 Letter match scores



	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	*
a	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
b	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	0	0	0	0	0
c	0	0	5	0	0	0	0	0	0	0	4	-2	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0
d	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	0	0	0	0	0
e	2	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
f	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	0	0	0	0	0
g	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	0	0	0	0	0
h	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
i	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
j	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
k	0	0	0	0	0	0	0	0	0	0	5	-2	0	0	0	0	0	0	-1	0	0	0	0	1	0	0	0
l	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	-1	0	0	0	0	0	0	0	0
m	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	-1	0	0	0	0	0	0	0	0
n	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	-1	0	0	0	0	0	0	0	0
o	1	0	0	0	0	0	-1	0	-0.1	0	0	0	0	0	5	0	0	0	0	0	2	0	0	0	0	0	0
p	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0
q	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0
r	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0
s	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	3	0
t	0	0	0	4	0	0	-1	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0
u	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0
v	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0
w	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	5	0	0	0	0
x	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0
y	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0
z	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0
*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

A-3 Abbreviations: phrases

`omg`	`oh my god`
`idk`	`I don't know`
`idek`	`I don't even know`
`omfg`	`oh my fucking god`
`wtf`	`what the fuck`
`smh`	`shaking my head`
`ily`	`I love you`
`ilysm`	`I love you so much`
`tbh`	`to be honest`
`idc`	`I don't care`
`stfu`	`shut the fuck up`
`btw`	`by the way`
`ik`	`I know`
`idgaf`	`I don't give a fuck`
`jk`	`just kidding`
`hmu`	`hit me up`
`bff`	`best friend`
`nvm`	`never mind`
`ffs`	`for fucks sake`

oomf one of my friends

A-4 Text correction example

```
Today stats: No new followers, One unfollower via
    http://t.co/KQMxmRhDsX
today starts: no new followers, one unfollowers
    via [url]
why doesn't accio exist in real life this is
    important
why doesn't action exist in real life this is
    important
I c ur picture ... but is that really you ??
i see your picture ... but is that really you ??
Straight a synoptic grateful dead towards strange
    circulating medium displacement: AuoZkEvxf
straight a sympathetic grateful dead towards
    strange circulating medium displacement:
    akgfjkdadutdfhsdfkfajf
Im Yeva and I eat poop
im yea and i eat poop
RT @Officialsimyai: If I have to beg fo yo
    attention den I don't want it
if i have to beg for you attention den i don't
    want it
@carinaa888 exactly!! sooo shut ur ass up lol
[username] exactly!! so shut your ass up [laughter
    ]
 @YungJadeTheGawd: I wanna have a cookout
    tomorrow .    In dere like swimwear
[username]: i wanna have a cookout tomorrow .
    in there like swimwear
RT @natalieben: Greens have got far more votes per
     minute of TV time than any other party.  #
    VoteGreen2014. #EP2014
greens have got far more votes per minute of tv
    time than any other party.  #votegreen2014. #
    ep2014
We all could die tonight
we all could die tonight
Back 2 Good
back 2 good
one person followed me // automatically checked by
     http://t.co/XTToHqv7gF
one person followed me // automatically checked by
     [url]
Why people be mad when someone runs to there ex?
    No reason for you tryna be a rebound anyway tf
     lol
why people be mad when someone runs to there ex?
    no reason for you tryna be a rebound anyway
    what the fuck [laughter]
RT @BestVinesEver: To a piano tho?! https://t.co/
    ISSmUHov2Q
to a piano tho?! [url]
RT @TheBucktList: Need it now.     http://t.co/
    PDXYKSKxfK
need it now.     [url]
@MikeyPosod hell yeah haha
[username] hell yeah  [laughter]
i get high off our memories
i get high off our memories
RT @CourtMitchell17: I feel like power dressing
    shouldnt be dressing like a man it should be
    owning our feminity &amp;wearing it proudly. I
     lov
i feel like power dressing shouldn't be dressing
    like a man it should be owning our feminist &
    amp;wearing it proudly. i love
Learn About Mountain Climbing from Novice to
    Experts  http://t.co/oky5Qsf1Df
learn about mountain climbing from notice to
    experts  [url]
@jdhowa2 Tuesday I typed "Waze is taking me the
    rapey way so if I'm not there in 10, let's
    hope my body shuts this whole thing down."
[username] tuesday i typed "wake is taking me the
    rape way so if i'm not there in 10, let's hope
     my body shuts this whole thing down."
Slim my new header               she so fine man

slim my new header               she so fine man

FOLLOW ME B  @JackDail 66
follow me be  [username] 66
@FotMound Can you give examples of a tightly
    focused theme? #fantasychat
[username] can you give examples of a tightly
    focused theme? #fantasychat
In love memory daddy http://t.co/OhBTgBZdP3
in love memory daddy [url]
My first jet lag...
my first jet lag...
RT @Nikaxo__: FUCK YALL THEY TRY !
fuck y'all they try !
RT @RomarioWay300: Who Else
```