Normalizing Social Media Texts by Combining Word Embeddings and Edit Distances in a Random Forest Regressor

Rob van der Goot
r.van.der.goot@rug.nl

28-05-2016
1 Problem

2 Error Detection

3 Generation

4 Ranking

5 Conclusion

6 Future Work
Adapt Natural Language Processing pipelines to noisy (web) data
Problem

- Adapt Natural Language Processing pipelines to noisy (web) data
- Normalize
Spelling Correction vs. Normalization

spell. corr.

normalization
Problem

Spelling Correction

abilites abilities

 teh the

kingdon kingdom
Problem

Normalization

abilites abilities
teh the
teh kingdom doing
kingdon doing
Bham Birmingham
doin
to
2

The There
ggggrrrrreeeeeeeaaaaaaatatttttt
great
ur your
Problem

Traditional spelling correction framework:
- Error detection
- Candidate generation
- Ranking of candidates
Problem

- Train set: 2,577 tweets from (Li and Liu 2014)
- Test set: LexNorm (Han and Baldwin 2011) 549 tweets
Outline

1. Problem
2. Error Detection
3. Generation
4. Ranking
5. Conclusion
6. Future Work
Error Detection

Spelling correction:
- Dictionary lookup
Error Detection

- Often skipped in normalization methods
- Here as well, because the goal is to be used in a pipeline
- All tokens are considered to be a possible error/disfluency
- Recall = 100%
- But the original word is always kept!
Outline

1. Problem
2. Error Detection
3. Generation
4. Ranking
5. Conclusion
6. Future Work
Spelling correction:
- Lexical edit distance
- Phonetic edit distance (Double Metaphone)
Spelling correction:
- Lexical edit distance
- Phonetic edit distance (Double Metaphone)
- Good results
Spelling correction:

- Lexical edit distance
- Phonetic edit distance (Double Metaphone)
- Good results
- So we use an existing system (Aspell)
Other disfluencies
- A more data aware model is necessary
Other disfluencies

- A more data aware model is necessary
- Semi-supervised
Generation

Other disfluencies

- A more data aware model is necessary
- Semi-supervised
- Word Embeddings
Word Embeddings

- Model taken from (Godin et al. 2015)
- Trained on 400 million Tweets
- 3,039,345 words
- Use cosine distance to find top-n words in vector-space
Generation

The graph illustrates the recall percentage of found corrections for different methods as a function of the number of candidates returned. The methods compared are Word Embeddings, Aspell, and their combination. The graph shows that as the number of candidates increases, the recall percentage also increases, with the combination method generally achieving the highest recall.
Outline

1. Problem
2. Error Detection
3. Generation
4. Ranking
5. Conclusion
6. Future Work
Ranking

Spelling correction:

- Combination of edit distances
Previous approaches:
- Ngram based approaches
- Combine Ranking with generation
My approach:

- Use features from generation
- Supplement these features with N-Gram features
- Google Ngrams \(^1\) & Twitter Ngrams \(^2\)
- Combine all features in a Random Forest Classifier
- Default parameters Scikit Learn, except for the number of trees \(= 100\)

---

\(^1\)Brants and Franz 2006
\(^2\)Herdağdelen 2013
Ranking

![Graph showing recall vs number of candidates returned]

- **Word Embeddings**
- **Aspell**
- **Google Unigrams**
- **Twitter Unigrams**
- **Random Forest**
Ranking (ablation)
### Ranking

<table>
<thead>
<tr>
<th>System</th>
<th>top1</th>
<th>top3</th>
<th>top10</th>
<th>top20</th>
<th>upper bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Li and Liu 2012)</td>
<td>73.0</td>
<td>81.9</td>
<td>86.7</td>
<td>89.2</td>
<td>94.2</td>
</tr>
<tr>
<td>(Li and Liu 2014)</td>
<td>77.14</td>
<td>86.96</td>
<td>93.04</td>
<td>94.82</td>
<td>95.90</td>
</tr>
<tr>
<td>(Li and Liu 2015)</td>
<td>87.58</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Our system</td>
<td>82.31</td>
<td>88.70</td>
<td>91.89</td>
<td>93.37</td>
<td>93.37</td>
</tr>
</tbody>
</table>
Outline

1. Problem
2. Error Detection
3. Generation
4. Ranking
5. Conclusion
6. Future Work
Conclusion

Overview

spell. corr.

normalization

Aspell

My approach
Conclusion

For the normalization task:

- Word embeddings complement edit distances well
- A random forest classifier works very well for ranking
- This is a simple system, with a reasonable performance
Outline

1. Problem
2. Error Detection
3. Generation
4. Ranking
5. Conclusion
6. Future Work
Future Work

- Multilingual/multiword embeddings
- Generation (build own language models)
Future Work

- Multilingual/multiword embeddings
- Generation (build own language models)
- Parameter tuning, add domain specific information
- Find candidate with: "word.*"
Future Work

- This system was created for use in a pipeline system
Future Work

- This system was created for use in a pipeline system
- Parse a word graph based on the output of this normalization