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

Outline

Problem

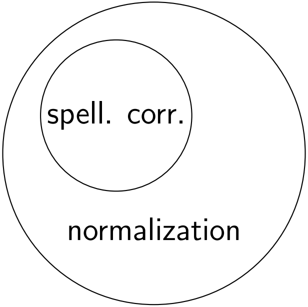
- Adapt Natural Language Processing pipelines to noisy (web) data

Problem

- Adapt Natural Language Processing pipelines to noisy (web) data
- Normalize

Problem

Spelling Correction vs. Normalization



Problem

Spelling Correction

abilites	abilities
teh	the
kingdon	kingdom

Problem

Normalization

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

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

Generation

Spelling correction:

- Lexical edit distance
- Phonetic edit distance (Double Metaphone)

Generation

Spelling correction:

- Lexical edit distance
- Phonetic edit distance (Double Metaphone)
- Good results

Generation

Spelling correction:

- Lexical edit distance
- Phonetic edit distance (Double Metaphone)
- Good results
- So we use an existing system (Aspell)

Generation

Other disfluencies

- A more data aware model is necessary

Generation

Other disfluencies

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

Generation

Other disfluencies

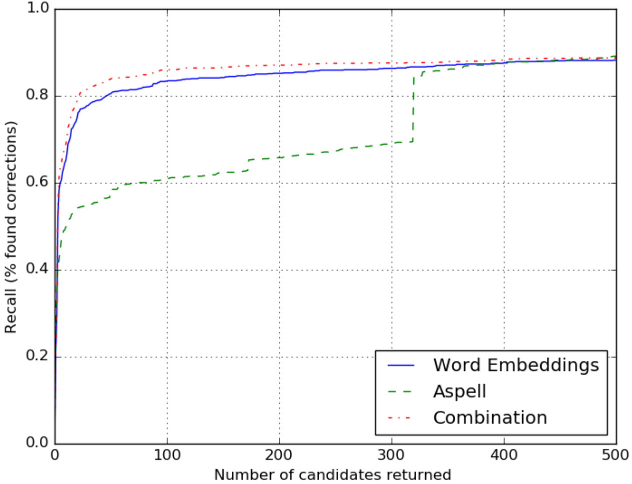
- A more data aware model is necessary
- Semi-supervised
- Word Embeddings

Generation

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



1.0
0.8
0.6
0.4
0.2
0.0
0
100
200
300
400
500
Recall (% found corrections)
Number of candidates returned
Word Embeddings
Aspell
Combination

Outline

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

Ranking

Spelling correction:

- Combination of edit distances

Ranking

Previous approaches:

- Ngram based approaches
- Combine Ranking with generation

Ranking

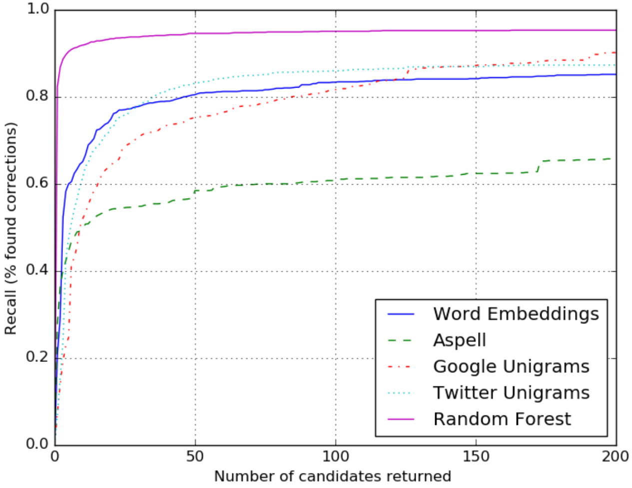
My approach:

- 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

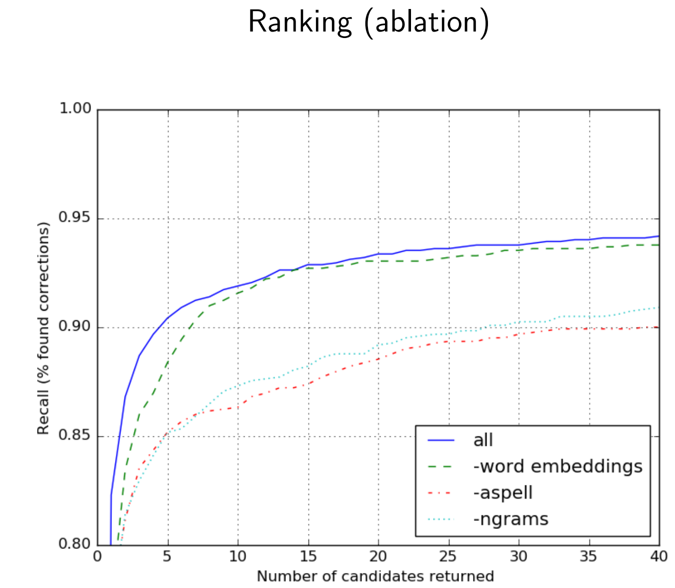
[2] Herdağdelen 2013

Ranking



1.0
0.8
0.6
0.4
0.2
0.0
Recall (% found corrections)
0
50
100
150
200
Number of candidates returned
Word Embeddings
Aspell
Google Unigrams
Twitter Unigrams
Random Forest

Ranking



Ranking (ablation)
1.00
0.95
0.90
0.85
0.80
0
5
10
15
20
25
30
35
40
Recall (% found corrections)
Number of candidates returned
all
-word embeddings
-aspell
-ngrams

Ranking

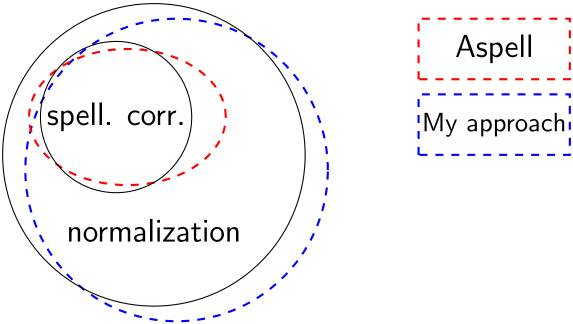
Ranking

System	top1	top3	top10	top20	upper bound
(Li and Liu 2012)	73.0	81.9	86.7	89.2	94.2
(Li and Liu 2014)	77.14	86.96	93.04	94.82	95.90
(Li and Liu 2015)	87.58				
Our system	82.31	88.70	91.89	93.37	93.37

Outline

Conclusion

Overview



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

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

Future Work

https://bitbucket.org/robvanderg/errcor