# Spelling correction with word and character n-gram embeddings

Simon Šuster
(joint work with Pieter Fivez and Walter Daelemans)

Computational Linguistics and Psycholinguistics
Research Center, University of Antwerp

# Subtasks in spelling correction

1. Detection of misspellings
2. Generation of replacement candidates
3. Ranking of candidates

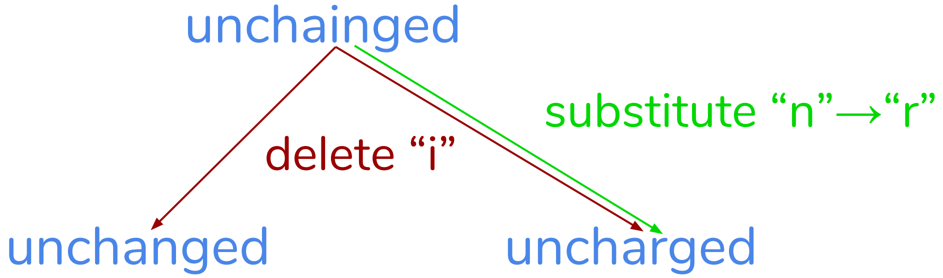
# 1. Detection of misspellings

- Non-word misspellings
  - Not a valid word of language vocabulary, e.g. “unchainged”
  - Use a vocabulary
  - Any out-of-vocabulary (OOV) word is marked as a misspelling
- Real-word misspellings
  - A valid word of the language, but not in the context: “[too] play a game”
  - Can’t rely on a vocabulary
  - Every word is a potential misspelling

# 2. Generation of replacement candidates

Create a set of orthographically or phonetically close words

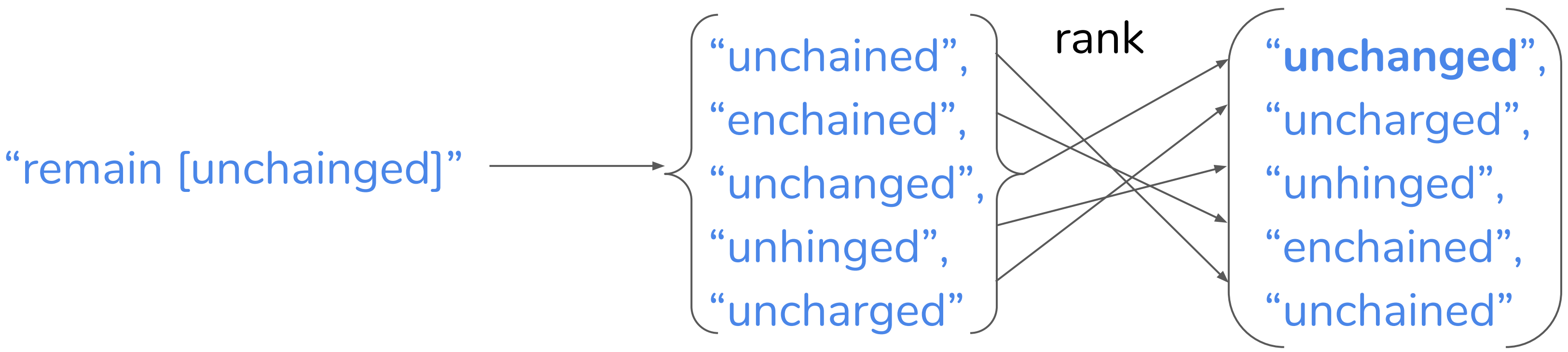
- Edit-distance measures
  - Operations: insertion, deletion, substitution
  - Allow candidates to be removed from the misspelling for a certain number of operations




- Can use surface forms or convert to a phonetic approximation:
  - “unchainged” → ANXNJT

# 3. Ranking of candidates

Pick the top-ranked word based on a scoring function



# Context-**insensitive** scoring

Assign the same correction regardless of the context:

- “Interest rates remain [unchainged]” → “unchanged”
- “[unchainged] melody” → “unchanged”

Typically based on an estimate of

- likelihood of character insertion, deletion and replacement operations
- prior unigram probability of the correct word

# Context-**sensitive** scoring

A context-sensitive model:

- “Interest rates remain [unchainged]” → “unchanged”
- “[unchainged] melody” → “unchained”

Prior: 2-gram, 3-gram, ... (word sequence) probabilities

- Works well when n is high (e.g. 5)
- Estimates must be obtained from very large corpora

# Noisy channel

- The scoring functions are examples of a noisy channel model
- Bayesian inference: see an observation (misspelling), find the word that generated it (correct)
- $p(\text{correct} \mid \text{misspelling}) = \underbrace{p(\text{misspelling} \mid \text{correct})}_{\text{likelihood}} * \underbrace{p(\text{correct})}_{\text{prior}}$

- A popular model in spelling correction

# Embeddings for spelling corrections

Spelling correction without the noisy-channel model

Context sensitivity without estimating the prior using longer n-grams

# Candidate scoring with vector semantics

$\text{cand}_i \longrightarrow \vec{\text{cand}}_i$

...

$\text{cand}_2 \longrightarrow \vec{\text{cand}}_2$

$\text{cand}_1 \longrightarrow \vec{\text{cand}}_1$

$w_{-3} \quad w_{-2} \quad w_{-1} \quad w_{\text{misspelled}} \quad w_{+1} \quad w_{+2} \quad w_{+3}$

sum into $\vec{w}$

$\text{score}_i = \text{cosine}(\vec{w}, \vec{\text{cand}}_i)$

Good for semantic compatibility, but ignores whether $w_{\text{misspelled}}$ and $\text{cand}_i$ are orthographically or phonetically similar

# Candidate scoring with vector semantics

$cand_i \longrightarrow \vec{cand}_i$

...

$cand_2 \longrightarrow \vec{cand}_2$

$cand_1 \longrightarrow \vec{cand}_1$

$w_{-3}$ $w_{-2}$ $w_{-1}$ $w_{misspelled}$ $w_{+1}$ $w_{+2}$ $w_{+3}$

sum into $\vec{w}$

$score_i = \text{cosine}(\vec{w}, \vec{cand}_i) * \text{weight}$

Solution: weight with Damerau-Levenshtein distance, Metaphone, or their combination

# Parameters

$cand_i \rightarrow \vec{cand}_i$

...

$cand_2 \rightarrow \vec{cand}_2$

$cand_1 \rightarrow \vec{cand}_1$

$\boxed{score_i = cosine(\vec{w}, \vec{cand}_i) * weight}$

$\underbrace{w_{-3} \quad w_{-2} \quad w_{-1}} \quad w_{misspelled} \quad \underbrace{w_{+1} \quad w_{+2} \quad w_{+3}}$

weighting function

window size

weight words differently

composition operation

sum into $\vec{w}$

# Experiments (Fivez et al., 2017)

Applied in the clinical domain for English and Dutch

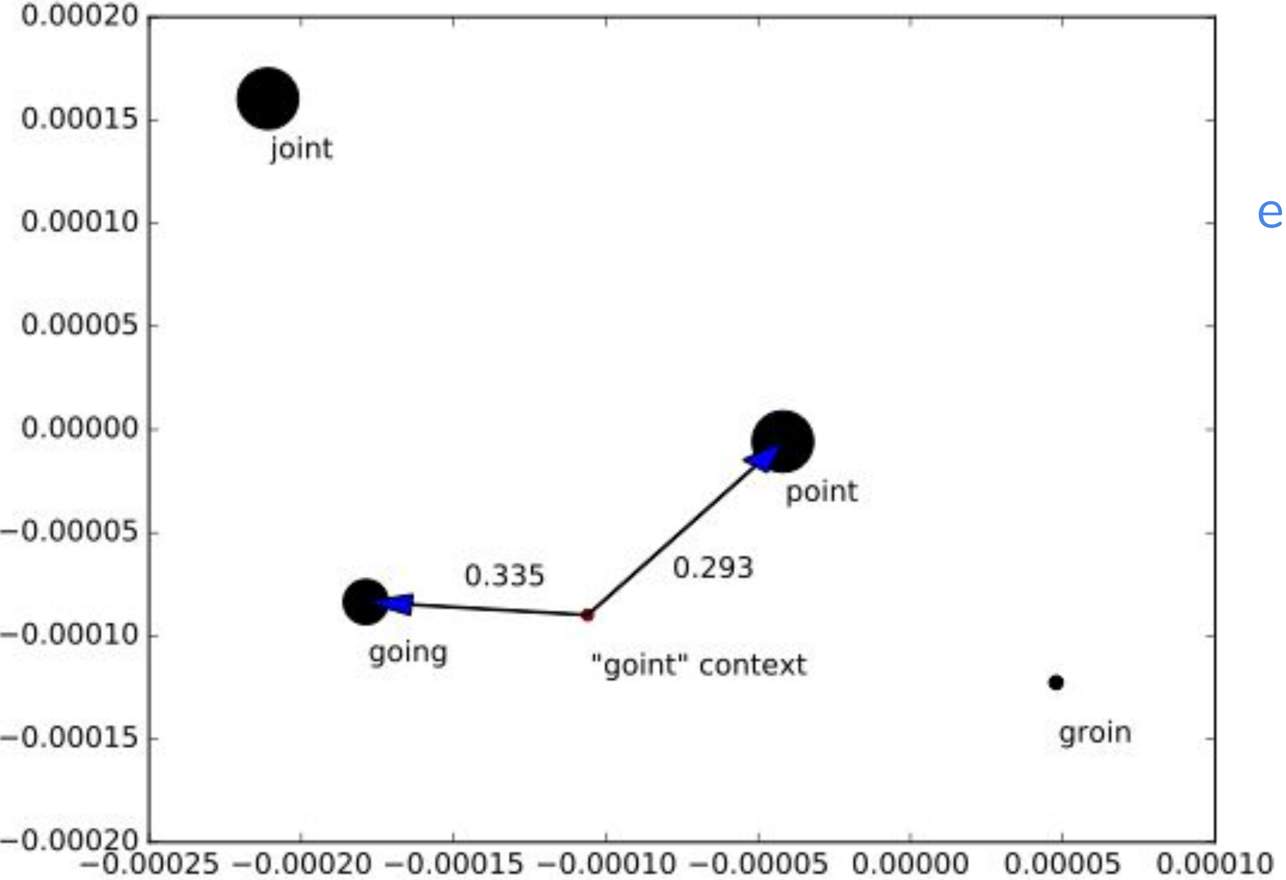
- Electronic health records (Antwerp University Hospital)
- Intensive care notes (Beth Israel Hospital)

- Model development on synthetic data
- Testing on human annotations (900 for EN, 500 for NL)
  - 88–90% accuracy

“sclerosin” → “sclerosing”
“sympots” → “symptoms”
“phebilitis” → “phlebitis”

“letels” → “letsels”
“wijnig” → “weinig”
“verminderderde” → “verminderde”

# Example of context sensitivity




"new central line lower extremity bypass with sob now [goint] to be intubated"

# Challenge

For our method to work well, we need:

- embedding for each candidate
  - but candidate may not be in the embedding vocabulary
- embedding for each context word
- embedding for the misspelling

How to represent OOV words with embeddings?

How to represent rare words with embeddings?

# Representing rare and OOV words

(assuming increasing the corpus size is not possible)

- assign a random vector (Dhingra et al., 2017)
- bin all rare words into a new “UNK” word type
- encode word definitions with an external resource (Long et al., 2016)
- train at morpheme level (Luong et al., 2013)
- train at the character n-gram level (Bojanowski et al., 2017)
    - nearest neighbors will also be more orthographically similar (character n-gram overlap)

# Learning of word embeddings

Achieving semantic similarity

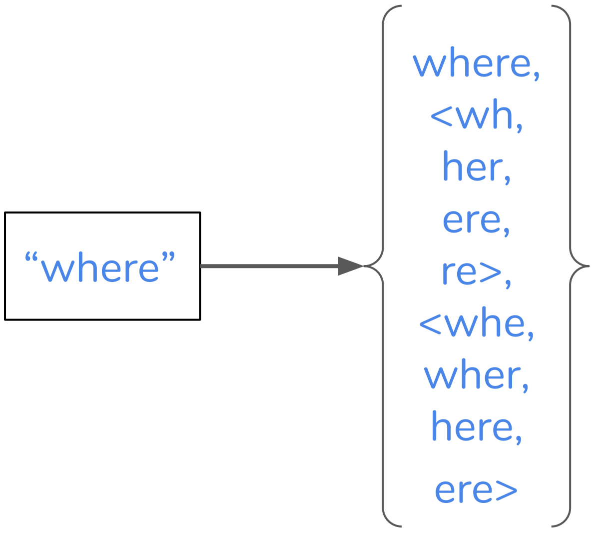
(Mikolov et al., 2013; aka **word2vec**)

- adjust weights of a classifier to best predict adjacent words
  - want $\boxed{\vec{w} \cdot \overrightarrow{context}}$ to be high
  - want $\boxed{\vec{w} \cdot \overrightarrow{negative}}$ to be low
- weights are embeddings
- 1 word = 1 embedding
- no knowledge of the internal word structure

# Learning of character n-gram embeddings

Achieving both semantic and spelling similarity
(Bojanowski et al., 2017; aka **fasttext**)

- Instead of word-only units:
  - add character n-grams of varying lengths
  - mark the beginning (“<”) and end (“>”) of words



# Learning of character n-gram embeddings

Achieving both semantic and spelling similarity
(Bojanowski et al., 2017; aka **fasttext**)

- Training objective is the sum of dot products between the target and the character n-grams
- At test time, the embedding is also obtained by summing

"where" → where, <wh, her, ere, re>, <whe, wher, here, ere>

Works especially well for languages
- with rich morphology (e.g. Slavic languages)
- rich with compound words (e.g. Dutch)

# Nearest neighbors:
# “delam” (EN: “to work”, 1st person sg.)

| | | |
|---|---|---|
| *“to finish working”* | oddelam 0.651 | 1. pers. sg. |
| *“to remake, to recycle”* | predelam 0.640 | 1. pers. sg. |
| *“to be doing (a job)”* | opravljam 0.617 | |
| *“to work”* | delajva 0.606 | 1. pers. dual, imperative |
| *“to create”* | ustvarjam 0.600 | |
| *“to process, to work on”* | obdelam 0.595 | 1. pers. sg. |
| *“to work”* | delajta 0.592 | 2. pers. dual, imperative |
| *“to create”* | izdelam 0.591 | 1. pers. sg. |
| *“to be active as”* | delujem 0.589 | |
| *“to work”* | delaš 0.586 | 2. pers. sg. |

*Source model: http://github.com/facebookresearch/fastText*

# Nearest neighbors for a rare word: “relmuis” (EN: “edible dormouse”)

| **word2vec** | **fasttext** |
|---|---|
| kafferbuffel 0.972 | woelmuis 0.945 |
| pimpelmees 0.971 | hazelmuis 0.938 |
| "Atheris 0.971 | eikelmuis 0.927 |
| "Protostega" 0.971 | huppelmuis 0.918 |
| "Conger 0.970 | bosmuis 0.913 |
| impala 0.970 | buideleikelmuis 0.911 |
| waterbok 0.970 | veeltepelmuis 0.908 |
| Laat-Siluur 0.969 | stekelmuis 0.905 |
| driedoornige 0.969 | bosspitsmuis 0.900 |
| haringhaai 0.969 | bosvleermuis 0.892 |

*Source corpus: wiki-nl, 230M tokens*

# “bosbees” (EN: ~”berry”)

The fasttext model does not really understand morphology (of course):

Spelling overlap, meaning further away

bosbeek 0.679
bosbeekjes 0.657
bosbesbij 0.627
bosbesuil 0.602
bosberg 0.596
bosbeekjuffers 0.588
bosbeschermer 0.579
bosbegroeiing 0.569
bosbeekschildpad 0.569
bosbeleid 0.568

# References


Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching Word Vectors with Subword Information. Transactions Of The Association For Computational Linguistics, 5, 135-146.

Dhingra, B., Liu, H., Salakhutdinov, R., & Cohen, W. W. (2017). A comparative study of word embeddings for reading comprehension. *arXiv preprint arXiv:1703.00993*.

Fivez, P., Šuster, S., & Daelemans, W. (2017). Unsupervised context-sensitive spelling correction of clinical free-text with word and character n-gram embedding. In 16th Workshop on Biomedical Natural Language Processing of the Association for Computational Linguistics (pp. 143-148).

Long, T., Lowe, R., Cheung, J. C. K., & Precup, D. (2016). Leveraging lexical resources for learning entity embeddings in multi-relational data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (short paper)*.

Luong, T., Socher, R., & Manning, C. (2013). Better word representations with recursive neural networks for morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning* (pp. 104-113).

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. In ICLR Workshop Papers