Neural String Edit Distance

Jindřich Libovický    Alexander Fraser

Presented at the 6th Workshop on Structured Prediction for NLP
May 27, 2022, Dublin
Outline

Levenshtein Distance

Neural Model

Cognate Detection

Transliteration & Grapheme-to-Phoneme
Levenshtein Distance
Black-box architectures vs. Levenshtein distance

- Char-level tasks use the same architectures as e.g., MT
- Overkill: large, hardly interpretable
- Levenshtein distance: transparent, interpretable...

...but weak and not flexible
We fix that!
### Levenshtein Distance Example

**Transcribe** \textbf{kitten} to \textbf{sitting}

|     | k | i | t | t | e | n |
|-----|---|---|---|---|---|---|
| 0   | 1 | 2 | 3 | 4 | 5 | 6 |
| s   | 1 | 1 | 2 | 3 | 4 | 5 | 6 |
| i   | 2 | 2 | 1 | 2 | 3 | 4 | 5 |
| t   | 3 | 3 | 2 | 1 | 2 | 3 | 4 |
| t   | 4 | 4 | 3 | 2 | 1 | 2 | 3 |
| i   | 5 | 5 | 4 | 3 | 2 | 2 | 3 |
| n   | 6 | 6 | 5 | 4 | 3 | 3 | 2 |
| g   | 7 | 7 | 6 | 5 | 4 | 4 | 3 |

- **empty string to empty string costs zero**
- **first column:** empty string \(\rightarrow\) sitting
- **first row:** delete kitten
- **substring** \textbf{kit} \(\rightarrow\) \textbf{sittin}
  - we got rid of \textit{ki} and have sitti - change \textit{t} \(\rightarrow\) \textit{n}  
    cost \(4 + 1 = 5\)
  - we have sitin and got rid of \textit{ki} - delete \textit{t}  
    cost \(5 + 1 = 6\)
  - already got rid of \textit{kit} and have sitin - add \textit{n}  
    cost \(3 + 1 = 4\) \(\leftarrow\) **minimum**
Problem of setting the operation costs

Transliteration from latin to cyrilics: Praha → Прага

- All characters are equivalent, but different UTF characters
- Either an expert can write the rules for the character costs
- Or we can try to learn the weights from data
Learnable Edit Distance (Ristad and Yianilos, 1998)

• Probabilistic formulation: one multinomial distribution over all possible operations
• Transcription probability (simple modification of the algorithm)
• Trained using **Expectation-Maximization** algorithm

More flexible: weights are estimated from the data

Rigid costs: do not depend on prefix or suffix
Neural Model
Main idea

Do the same thing...

...and backpropagate the objective into a contextualized neural representation.
Model

• Get contextualized representation of input characters
• Symbol pairs: concatenate their representation and apply projection
• Estimate the insert, delete and substitute operations probabilities from these representations
The original EM algorithm assumes a **discrete operation table**...  
...but we have **continuous representations**.

- Expected distribution (forward-backward algorithm) – compared to actual distribution — optimize **KL divergence** between the predicted and expected distribution
- Directly optimize task-specific loss:
  - *String-pair classification*: optimize classification likelihood
  - *String transduction*: optimize output symbol negative log likelihood
Cognate Detection
For a pair of IPA strings...

ˈzɛlɛnːi:  zɛˈɫɛnɪj  ✓
ˈɦrubiː pyknós  ×
tu tam ✓

...decide if they have the same diachronic origin.

- Databases for Indo-European and Austro-Asiatic languages (Rama et al., 2018)
- Sampled positive and negative pairs, F1-measure for hits
- Use neural string edit distance to estimate the cognate probability
Example: Scores in the dynamic programing table

Cognate

Non-cognate
## Results

| Method                        | # param. | Indo-European | Austro-Asiatic |
|-------------------------------|----------|---------------|---------------|
|                               |          | $F_1 \uparrow$ | Time          | $F_1 \uparrow$ | Time          |
| Learnable edit distance       | 0.2M     | 32.8          | 0.4h          | 10.3           | 0.2h          |
| Transformer [CLS]             | 2.7M     | 93.5          | 0.7h          | 78.5           | 0.6h          |
| STANCE RNN                    | 1.9M     | 80.6          | 0.3h          | 16.7           | 0.2h          |
| ours                          |          |               |               |                |               |
| unigram                       | 0.5M     | 80.1          | 1.5h          | 48.4           | 0.7h          |
| CNN (3-gram)                  | 0.7M     | 93.9          | 0.9h          | 77.9           | 0.5h          |
| RNN                           | 1.9M     | 97.1          | 1.9h          | 84.0           | 1.2h          |
Transliteration & Grapheme-to-Phoneme
String Transduction Tasks

**Arabic → English Transliteration**

- 13k training, 1.5k validation and testing (Rosca and Breuel, 2016)

| Arabic     | English |
|------------|---------|
| ساندي     | sandy   |
| دايي       | daye    |
| ساروني     | saronni |
| أبركرميي   | abercromby |
| كورت       | kurt    |

**Grapheme-to-Phoneme Conversion**

- CMUDict dataset (Weide, 2005)
- 108k training, 5k valid., 13k test
- Multiple transcriptions, during evaluation, choose the closest one

| String      | Transcription |
|-------------|---------------|
| PERRON      | P EH R AH N   |
| TABUCHI     | T AA B UW CH IY |
| CUVELIER    | K Y UW V L IY ER |
| CONSUMERS’  | K AH N S UW M ER Z |
| KINGDOMS    | K IH NG D AH M Z |

Evaluation with Word Error Rate (WER) and Character Error Rate (CER)
Model modifications

• Unidirectional representation of the target
• Deletion probability must not depend on the last target character
• Dirty trick: Added attention from the target representation to source representation
| Method         | # Param. | CER | WER | Time |
|---------------|---------|-----|-----|------|
| RNN Seq2seq   | 3.3M    | 22.0 | 75.8 | 12m  |
| Transformer   | 3.1M    | 22.9 | 78.5 | 11m  |
| ours unigram  | 0.7M    | 31.2 | 85.0 | 36m  |
| CNN 3-gram    | 1.1M    | 24.5 | 80.1 | 41m  |
| ours RNN      | 2.9M    | 22.0 | 77.4 | 60m  |
## Results: Grapheme-To-Phoneme

| Method         | # Param. | CER↓ | WER↓ | Align.↑ | Time  |
|----------------|----------|------|------|---------|-------|
| RNN Seq2seq    | 3.3M     | 3.5  | 23.6 | 24.5    | 1.8h  |
| Transformer    | 3.1M     | 6.5  | 26.6 | 33.2    | 1.1h  |
| ours           |          |      |      |         |       |
| unigram        | 0.7M     | 20.6 | 66.3 | 59.5    | 2.4h  |
| CNN 3-gram     | 1.1M     | 12.8 | 48.4 | 38.1    | 2.5h  |
| RNN            | 2.9M     | 7.3  | 31.9 | 38.9    | 2.3h  |
Summary

• Generalized learnable edit distance for neural representations
• Can be used for string-pair classification and string transduction
• Competitive performance, better interpretability
Taraka Rama, Johann-Mattis List, Johannes Wahle, and Gerhard Jäger. Are automatic methods for cognate detection good enough for phylogenetic reconstruction in historical linguistics? In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Volume 2 (Short Papers), pages 393–400, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-2063. URL https://aclanthology.org/N18-2063.

Eric Sven Ristad and Peter N. Yianilos. Learning string-edit distance. IEEE Trans. Pattern Anal. Mach. Intell., 20(5):522–532, 1998. doi: 10.1109/34.682181. URL https://doi.org/10.1109/34.682181.

Mihaela Rosca and Thomas Breuel. Sequence-to-sequence neural network models for transliteration. CoRR, abs/1610.09565, 2016. URL http://arxiv.org/abs/1610.09565.

Robert Weide. The Carnegie-Mellon pronouncing dictionary [cmudict. 0.7]. Pittsburgh, PA, USA, 2005. Carnegie Mellon University. URL http://www.speech.cs.cmu.edu/cgi-bin/cmudict.