Spelling Correction and Autocompletion for Mobile Devices

Ziang Lu
Bachelor's Thesis

Computer Science Department
University of Freiburg

Examiner: Prof. Dr. Hannah Bast
Adviser: Matthias Hertel
Motivation

Number of smartphone users from 2016 to 2026 (in billions)

(Statista, 2021)
**Tasks**

we are going to watch a movie

- **correct the next word**
  
  current input:  
  
  candidates to choose: We Lee Bee

- **predict the next word**
  
  current input: We (expecting prediction)
  
  candidates to choose: are do were

- **complete the next word**
  
  current input: We are g (expecting completion)
  
  candidates to choose: going gone getting
Implementation and usage

Implementation as an Android System Keyboard

- choose candidates from vocabulary according to similarity
- rank candidates and show the best three candidates to the user
Prefix Edit Distance

We input always from the start of a word

Edit Distance (som, something) = 6

Prefix Edit Distance (PED):

\[ PED(x, y) = \min_{y'} (ED(x, y')) \]

where \( y' \) is a prefix of \( y \)
PED

PED(same, something) = ?

Given a threshold of PED and prefix “som”
Q-gram Index

Filter impossible candidates to accelerate response
Some words are very short like “I”, “am”, “a”

Pad special symbols: “$$I$$”, “$$am$$” ...

\[
\text{comm}(x', y') \geq |x| - q \cdot \delta
\]

\[
x' = $$freiburg
\]
\[
y' = $$breiberg
\]

\[
\text{comm}(x', y') = 2 \geq |8| - 3 \cdot 2.
\]
N-gram model

sequence of n words

Example:

h: have you seen my
w: book

\[ P(\text{book}|\text{my}) = \frac{C(\text{my book})}{C(\text{my})} \]

based on bigram

\[ P(\text{book}|\text{seen my}) = \frac{C(\text{seen my book})}{C(\text{seen my})} \]

based on trigram
N-gram probability

Extend the formula to N-gram:

\[ P(w_n|w_{n-N+1}^{n-1}) = \frac{C(w_{n-N+1}^{n-1} w_n)}{C(w_{n-N+1}^{n-1})} \]

(n represents the length of a whole sentence. \( w_J^I \) represents a sequence which consists of \( w_I, w_{I+1}, w_{I+2}, \ldots, w_J \).)

Combination of different N-gram probabilities:

\[ P(\text{N-gram}) = \sum_{i=1}^{N} P(w_n|w_{n-N+1}^{n-1}) \cdot \lambda_i \]
**POS-tags**

For absence of some combinations:

```
I have a cat
```

**Example:**

“she sleeps” never appears in the corpus

candidates: sleeping **sleeps** sleep
Corpus and Vocabulary

9000 words from MIT word list + 1000 most frequent words from corpus

14 million words make up of the corpus, which was retrieved from web texts.
Data storage

- avoid reading data on startup => store n-grams and q-gram index in a database
- load data only on demand
- lower delay of startup and low memory footprint
Methods and test set

**Testset**: 5% of corpus and Enron Emails

**Criterion** of evaluation: number of saved keystrokes (%)

Example:

sentence: Let’s have a drink.

length (including space): 19

keystrokes needed with keyboard: 10

\[
\frac{19 - 10}{19} \approx 47.37\% 
\]
Results and analysis

Performance based on different grams

<table>
<thead>
<tr>
<th>Gram</th>
<th>Saved keystrokes(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>bigram</td>
<td>39.30%</td>
</tr>
<tr>
<td>trigram</td>
<td>39.97%</td>
</tr>
<tr>
<td>quadrigram</td>
<td>39.98%</td>
</tr>
<tr>
<td>quinquegram</td>
<td>39.98%</td>
</tr>
</tbody>
</table>
**Test for the POS-Tags**

Evaluate the performance with the attendance of POS-Tags

<table>
<thead>
<tr>
<th></th>
<th>Saved keystrokes(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Without POS-Tags</td>
<td>39.31%</td>
</tr>
<tr>
<td>With POS-Tags</td>
<td>39.98%</td>
</tr>
<tr>
<td>+</td>
<td>0.67%</td>
</tr>
</tbody>
</table>

*She sleeps very well and he walks into the forest*
Test for the correction

The first letter of every word will be changed

100 sentences from Enron Emails

<table>
<thead>
<tr>
<th>Keyboard</th>
<th>No mistake</th>
<th>With mistakes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gboard</td>
<td>48.81%</td>
<td>19.84%</td>
</tr>
<tr>
<td>Nboard</td>
<td>45.64%</td>
<td>21.57%</td>
</tr>
</tbody>
</table>
Current problems and Future work

Problems:

● POS-tagger was not precise
● Paid little attention to more parts of a sentence

TODOs:

● customized and preciser POS-tagger
● more complicated language model => Transformer
Conclusion

- Theories and their application
- Implementation of the keyboard
- Evaluation
Thanks

Any other questions?