Spell Checker

Levenshtein Distance

Cedric Tsatia

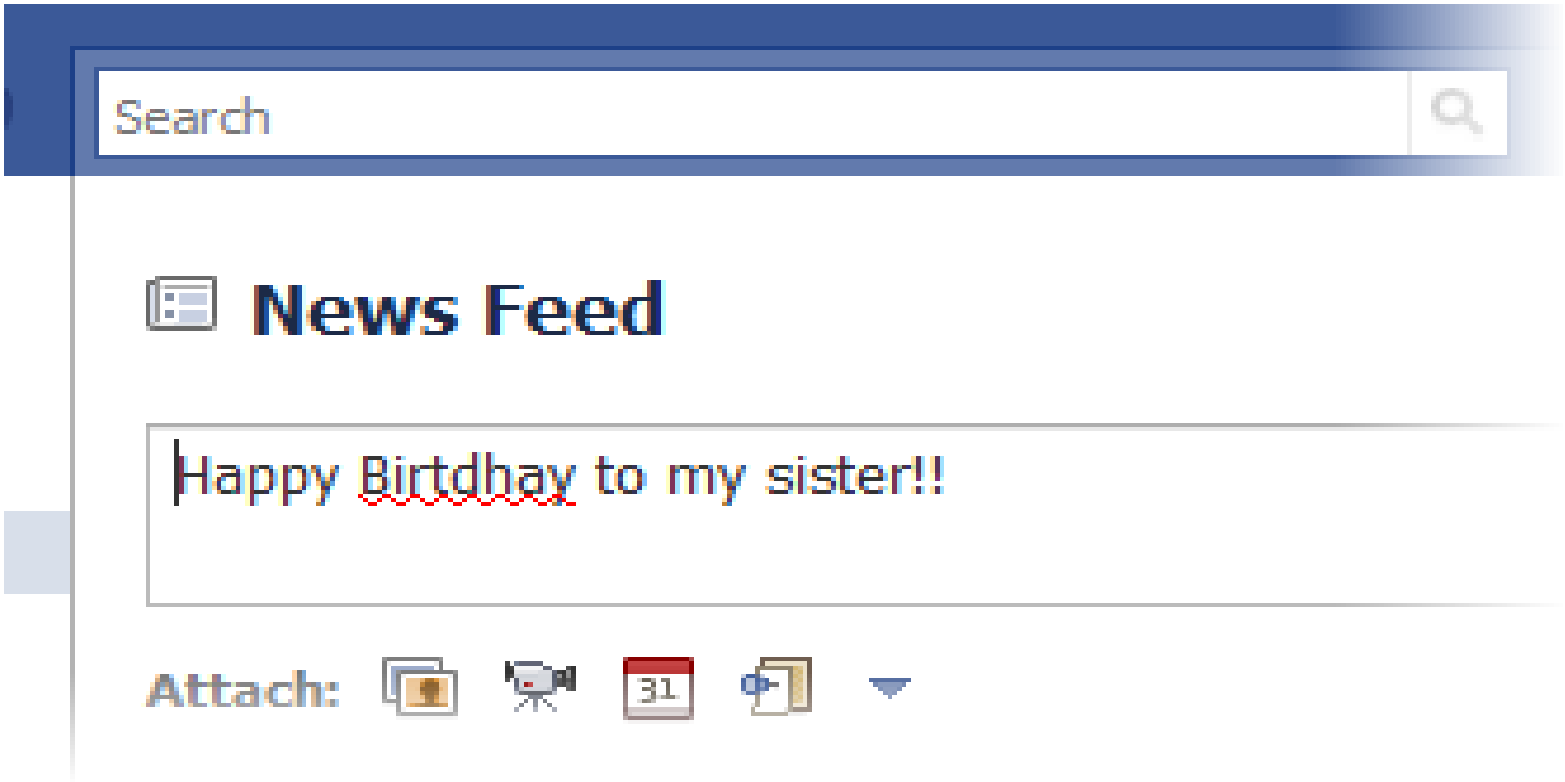
Department of Computer Science
University of Kiel

February 6, 2019

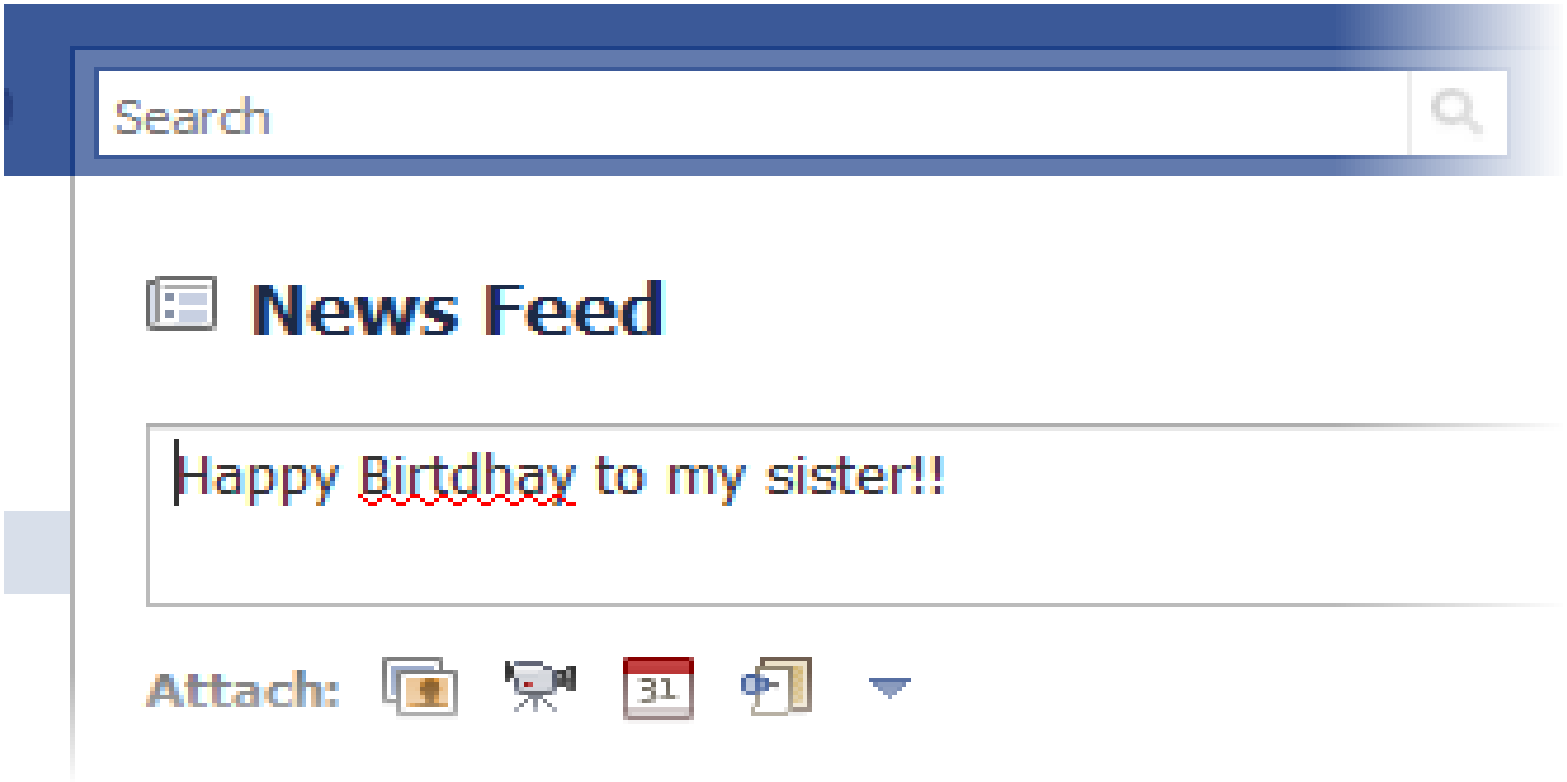
Definition(Wikipedia)

In software, a spell checker (or spell check) is a software feature that checks for misspellings in a text. Features are often in software, such as a word processor, email client, electronic dictionary, or search engine.

Facebook Spell Checker



Facebook Spell Checker



Problem → red wavy line

How to solve this Problem?

Is the underline string a "word"?

How to solve this Problem?

Is the underline string a "word"?

- Break string up into "words"?

How to solve this Problem?

Is the underline string a "word"?

- Break string up into "words"?
- Look up each word in a dictionary

How to solve this Problem?

Is the underline string a "word"?

- Break string up into "words"?
- Look up each word in a dictionary
- If found, then a word (no red Wavy Line)

How to solve this Problem?

Is the underline string a "word"?

- Break string up into "words"?
- Look up each word in a dictionary
- If found, then a word (no red Wavy Line)
- Else, not a word :(

Actually more complex ...

- would the dictionary store all variations of a word?
  - nice, niece, niche, Nice

Actually more complex ...

- would the dictionary store all variations of a word?
  - nice, niece, niche, Nice
- Some words are only correct if they have proper capitalization.
  - Kiel vs. kiel

Actually more complex ...

- would the dictionary store all variations of a word?
  - nice, niece, niche, Nice
- Some words are only correct if they have proper capitalization.
  - Kiel vs. kiel
- Some words have space in them.
  - au pair, et cetera

Spelling Suggestions

```
using System;
namespace ConsoleApplication5
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World, nicce to meet you");
        }
    }
}
```

Hold Ctrl to replace all
nice
niece
niche
Nice
Ignore Once
Ignore All
Add to Dictionary

How it works...

From all possible "correction candidates" of a word get those with the highest probability for the suggestions box.

How it works...

From all possible "correction candidates" of a word get those with the highest probability for the suggestions box.

We are trying to find the correction c, out of all possible candidate corrections(nice, niece, niche, Nice), that maximizes the probability that c is the intended correction, given the original word w:
$\text{argmax}_{c \in candidates} P(c|w)$

How it works...

From all possible "correction candidates" of a word get those with the highest probability for the suggestions box.

We are trying to find the correction c, out of all possible candidate corrections(nice, niece, niche, Nice), that maximizes the probability that c is the intended correction, given the original word w:
$\text{argmax}_{c \in candidates} P(c|w)$

Bayes Theorem

$\text{argmax}_{c \in candidates} P(w|c)P(c)/P(w)$
Since P(w) is the same for every possible candidate c, we can factor it out.
$\text{argmax}_{c \in candidates} P(w|c)P(c)$

Summary

Selection Mechanism: argmax

We choose the candidate with the highest combined probability.

Summary

Selection Mechanism: argmax

We choose the candidate with the highest combined probability.

Candidate Model: c candidates

This tells us which candidate corrections, c, to consider.

Summary

Language Model: P(c)

The probability that c appears as a word of English text. For example, occurrences of "the" make up about 7% of English text, so we should have P(the) = 0.07.

Summary

Language Model: P(c)

The probability that c appears as a word of English text. For example, occurrences of "the" make up about 7% of English text, so we should have P(the) = 0.07.

Error Model: $P(w|c)$

The probability that w would be typed in a text when the author meant c. For example, $P(teh|the)$ is relatively high, but $P(theeexyz|the)$ would be very low.

But how do we get the correction candidates?

Edit Distance

Edit Distance is a way of quantifying how dissimilar two strings (e.g., words) are to one another by counting the minimum number of Levenshtein distance operations required to transform one string into the other.

But how do we get the correction candidates?

Edit Distance

Edit Distance is a way of quantifying how dissimilar two strings (e.g., words) are to one another by counting the minimum number of Levenshtein distance operations required to transform one string into the other.

Levenshtein distance operations

- Insertion
- Deletion
- Substitution

But how do we get the correction candidates?

Edit Distance

Edit Distance is a way of quantifying how dissimilar two strings (e.g., words) are to one another by counting the minimum number of Levenshtein distance operations required to transform one string into the other.

Levenshtein distance operations

- Insertion
- Deletion
- Substitution

Brute force Algorithm ...

Complexity

The time complexity of this solution is exponential. In worst case, we may end up doing $O(3^{max(m,n)})$ operations. The worst case happens when none of characters of two strings match.
Worst case example str1 = "abc", str2 = "def"

Complexity

The time complexity of this solution is exponential. In worst case, we may end up doing $O(3^{max(m,n)})$ operations. The worst case happens when none of characters of two strings match.
Worst case example str1 = "abc", str2 = "def"

Cause of "bad" Complexity

We can see that many subproblems are solved, again and again, for example, eD(2,2) is called three times(See whiteboard). Since same suproblems are called again, this problem has Overlapping Subprolems property.

Complexity

The time complexity of this solution is exponential. In worst case, we may end up doing $O(3^{max(m,n)})$ operations. The worst case happens when none of characters of two strings match.
Worst case example str1 = "abc", str2 = "def"

Cause of "bad" Complexity

We can see that many subproblems are solved, again and again, for example, eD(2,2) is called three times(See whiteboard). Since same suproblems are called again, this problem has Overlapping Subprolems property.

Solution

Dynamic Programming...

How to proceed...

Complexity

$O(m * n)$ where m and n are the length of the words.

How to proceed...

Complexity

$O(m * n)$ where m and n are the length of the words.

- If a word is not in the dictionary then generate the set of all words having edit distance 1 to original word.
  - Use Bayes Theorem to get the "best correction".

How to proceed...

Complexity

$O(m * n)$ where m and n are the length of the words.

- If a word is not in the dictionary then generate the set of all words having edit distance 1 to original word.
  - Use Bayes Theorem to get the "best correction".
- Else generate the set of all words with edit distance 2 to the original word.
  - Use Bayes Theorem to get the "best correction".

How to proceed...

Complexity

$O(m * n)$ where m and n are the length of the words.

- If a word is not in the dictionary then generate the set of all words having edit distance 1 to original word.
  - Use Bayes Theorem to get the "best correction".
- Else generate the set of all words with edit distance 2 to the original word.
  - Use Bayes Theorem to get the "best correction".
- Otherwise consider the original word, even though it is not known or ask the user to add it to dictionary.