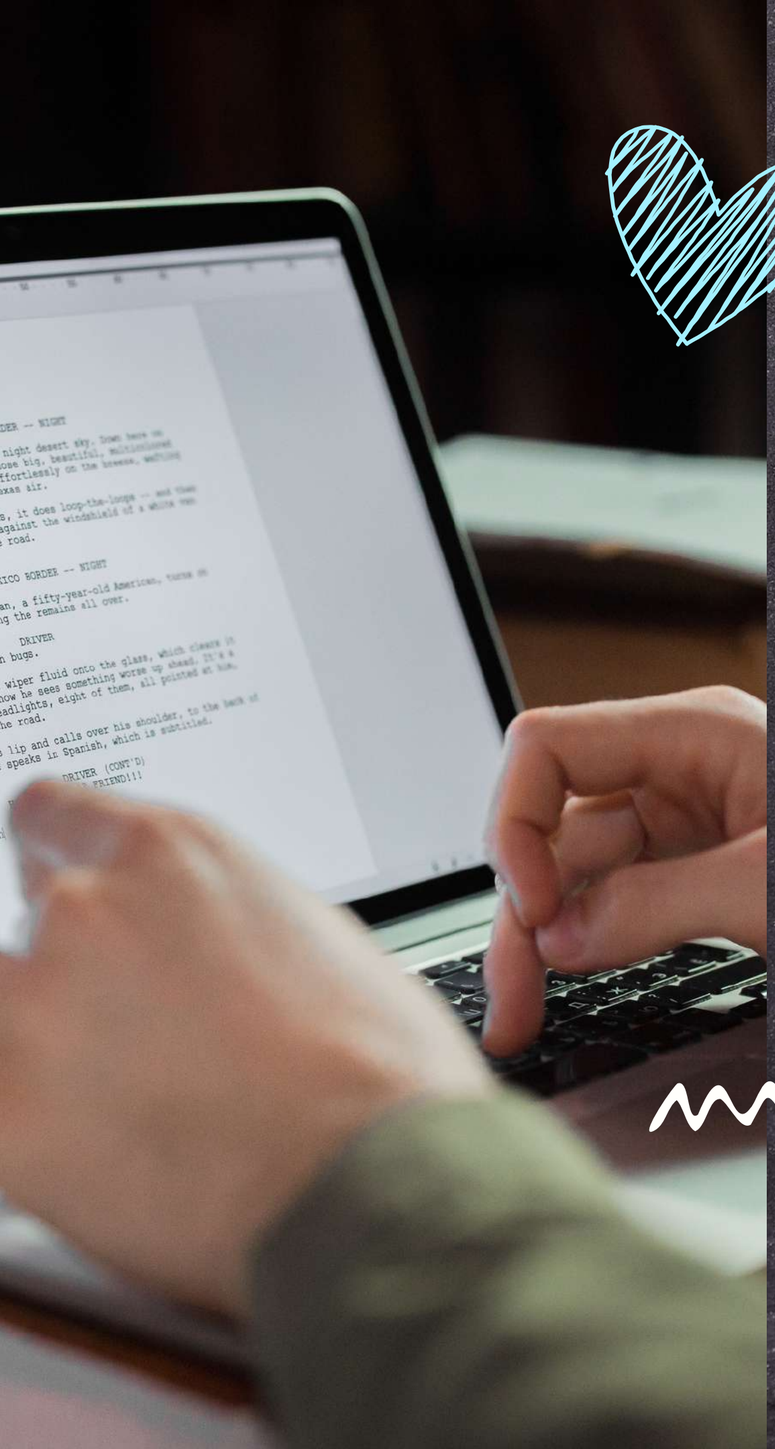
TYPO DETECTOR & CORRECTOR

2540096182 - ALEXANDER THEODORE
2540125131 - DENNIS LIM KAM HO
2540096081 - JOVIAN CHRISTOPHER HIZKIA
2540123542 - VINCENTIUS
2540123510 - HANS REGINALD



Background

- Typo detector dapat membantu mencari kata dan mengoreksi kesalahan penulisan yang dapat mengurangi kualitas teks
- Typo detector dapat mengetahui posisi kesalahan dalam sebuah kata dan memberikan saran kata yang lebih tepat
- Typo detector dapat meningkatkan efisiensi dan kenyamanan dalam menulis teks

Contoh

- Dalam penelitian Konchady et al. (2009), typo detector dengan n-gram dapat mendeteksi dan mengoreksi kesalahan penulisan dalam bahasa Inggris dengan menghasilkan tingkat akurasi yang tinggi diatas 85%

Source:

- Konchady, M. (2009). Detecting Grammatical Errors in Text using a Ngram-based Ruleset. ResearchGate. https://www.researchgate.net/publication/255654796_Detecting_Grammatical_Errors_in_Text_using_a_Ngram-based_Ruleset

https://www.researchgate.net/publication/255654796_Detecting_Grammatical_Errors_in_Text_using_a_Ngram-based_Ruleset

Related Works

Algoritma ini sudah diterapkan oleh beberapa proyek yang dibuat oleh orang lain. Contohnya adalah:

- Reverso Spell Checker : https://www.reverso.net/spell-checker/english-spelling-grammar
- Quillbot Spell Checker : https://quillbot.com/spell-checker
- Grammarly Spell Checker : https://www.grammarly.com/spell-checker

Grammar Checker & Rephraser

Check spelling, grammar and style for English texts

English Copy

In a groundbreaking study, researchers at the renowned Life Sciences Institute have made a significant breakthrough in the field of medical treatment, offering new hope for patients around the world. The study, led by Dr. Alexnder Johnson, focused on developing a novel therapy for neurological disorders, particularly Alzheimer's disease.

Back Edit Translate Rephrase New

343 characters, 49 words | 7 auto-corrected mistakes

NEW

Refine your style with our

REPHRASER

QuillBot

Spell Checker

Upgrade to Premium

QuillBot for Chrome | Write like a pro, everywhere you write.

Add to Chrome. It's free!

English (US) German French Spanish

Paragraph

In a groundbreaking study, researchers at the renowned Lfie Sciences Institute have made a significant btreakthrough in the field of medical traeatment, offering new hope for patietns around the world. The study, led by Dr. Alexnder Johnson, focused on developing a novev therapy for neurologiucal disorders, particularly Alzheimer's diseaese.

49 Words • 6 Errors

Fix All Errors

Paraphrase Text

tiny

In a groundbreaking study, researchers at the renowned Lfie Sciences Institute have made a significant btreakthrough in the field of medical traeatment, offering new hope for patietns around the world. The study, led by Dr. Alexnder Johnson, focused on developing a novev therapy for neurologiucal disorders, particularly Alzheimer's diseaese.

7 Suggestions

Let's get started.

Step 1: Add your text, and Grammarly will underline any issues.
Step 2: Hover over the underlines to see suggestions.
Step 3: Click a suggestion to accept it.

Get Grammarly It's free

Already have an account? Log in

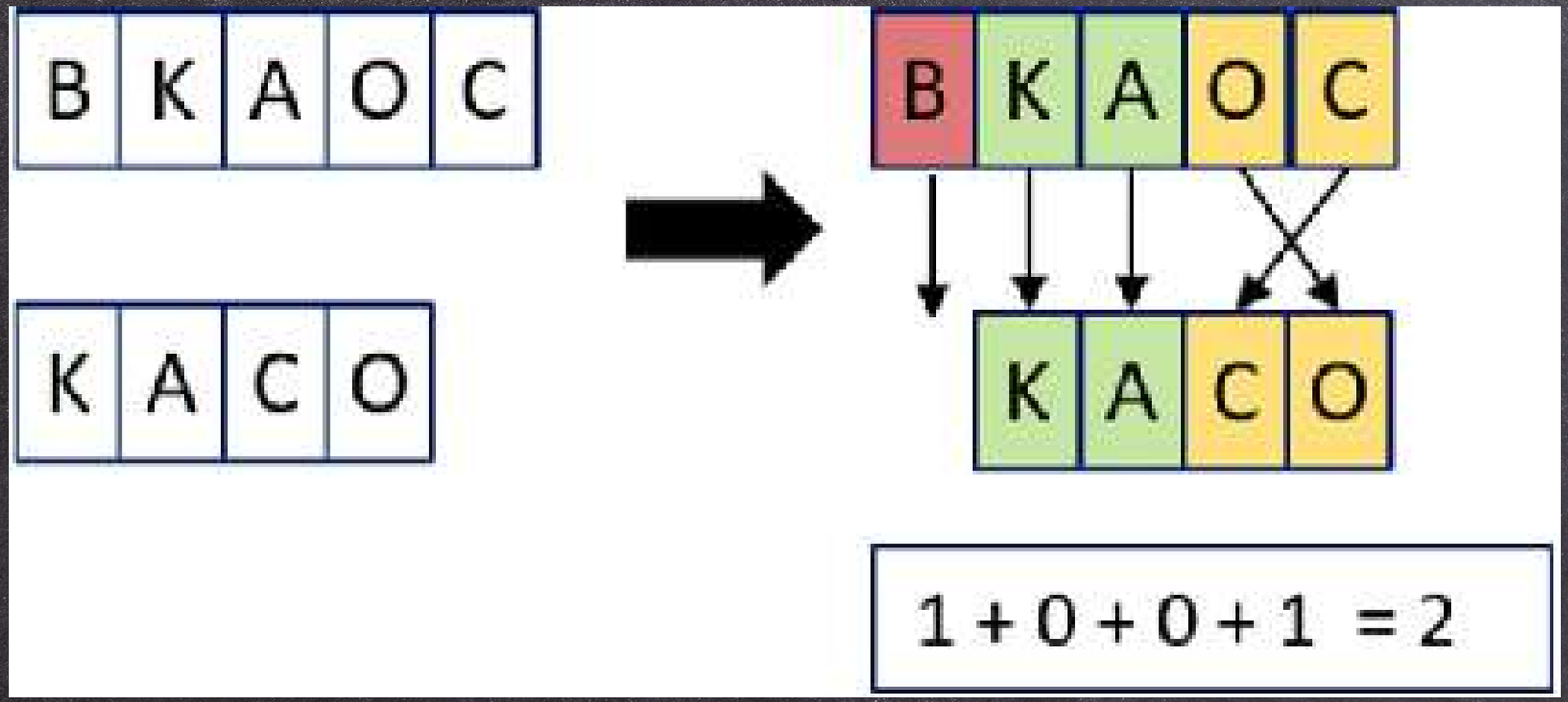
Related Works

- FASPell: A Fast, Adaptable, Simple, Powerful Chinese Spell Checker Based On DAE-Decoder Paradigm oleh Yuzhong Hong, Xianguo Yu, Neng He, Nan Liu, Junhui Liu (2019)
- Spell Checker oleh Vibhakti V. Bhaire, Ashiki A. Jadhav, Pradnya A. Pashte, Mr. Magdum P.G (2015)
- Spell checker for consumer language (CSpell) oleh Chris J Lu, Alan R Aronson, Sonya E Shooshan, Dina Demner-Fushman (2019)

Methods

Algoritma yang dipakai pada proyek kelompok kami adalah:

- Membuat fungsi edit distance (Damerau-Levenshtein Distance)
- Membuat class ngram checker, dihitung berdasarkan probabilitas paling tinggi
- Mencari data "training" lewat "fitting" class tadi terhadap korpus reuters, built in dari library nltk
- Menguji model terhadap dokumen dengan tipo lalu diuji akurasinya dengan dokumen tanpa tipo



Methods

Model

```python
# Bikin class biar rapih
class ngram_checker:
    # Kasih text training sama angka n_gram yang diinginkan.
    def __init__(self, text:int, n_gram:int):
        self.n_gram = n_gram
        self.word_freq = self._get_freq(text, n_gram)
        self.vocab = set(self.word_freq["word"].apply(lambda x: x[0]).unique())

    # Generate df ngram, dengan padding
    def _get_freq(self, text:str, n_gram:int):
        token = word_tokenize(text)
        grams = ngrams(token, n_gram, pad_left = True, pad_right = True, left_pad_symbol="<s>", right_pad_symbol="</s>")
        df = pd.DataFrame(Counter(grams).items(), columns=["word", "freq"])
        return df

    # Dari sebuah list of tokens dimana elemen terakhirnya typo,
    # buat daftar frekuensi huruf2 yang jaraknya 2 edit distance
    # aka token pengganti.
    def prob_grams(self, tokens:list[str]):
        ret = pd.DataFrame(columns=self.word_freq.columns)

        # Atur padding
        if len(tokens) < self.n_gram:
            tokens = (["<s>"] * (self.n_gram - len(tokens))) + tokens
        else:
            tokens = tokens[-self.n_gram:]

        # Ret kalo ada di vocab, bukan typo.
        word = tokens[-1]
        if word in self.vocab:
            return ret

        # Generate semua string yang mungkin buat edit distance 2, terus pilih yang dalam vocab kita.
        possible_edits = edits2(word)
        vocab_words = possible_edits.intersection(self.vocab)

        # filter dulu yang depannya sesuai, biar kodenya lebih cepet
        freq = self.word_freq[self.word_freq["word"].apply(lambda x: x[:-1] == tuple(tokens[:-1]))]
```

Methods

```python
    # sekarang filter yang belakangnya sesuai
    for word in vocab_words:
        ret = pd.concat((ret, freq[freq["word"].apply(lambda x: x[-1] == word)]))

    return ret.sort_values("freq", ascending=False)

# Fungsi yang bakal ngejalanin prob_grams secara otomatis.
# Bakal nerima string habis itu secara otomatis tokenize.
# Habis ditokenize diproses pake fungsi atas buat dapet list2 token pengganti.
# Ambil token pengganti dengan probabilitas tertinggi, terus sambung.
def autocorrect(self, text:str):
    # Hasil autocorrect
    new_tokens = []

    # Hubungin kata yang satu dengan yang lain
    tokens = word_tokenize(text)
    ngram_tokens = list(ngrams(tokens, self.n_gram, pad_left=True, left_pad_symbol = "<s>"))
    # Bikin list of list biar bisa dimutate
    ngram_tokens = [list(l) for l in ngram_tokens]

    # Jalanin
    for t in ngram_tokens:
        result = self.prob_grams(t)
        if len(result) == 0:
            new_tokens.append(t[-1])
        else:
            print(f"Corrected {t} to: ", end="")
            t[-1] = result.iloc[0]["word"][-1]
            new_tokens.append(t[-1])
            print(f"{t}")
    return TreebankWordDetokenizer().detokenize(new_tokens)
```

Methods

```python
[ ] # Kalo trigram ketemu, pake punya trigram, kalo gak pake punya bigram, kalo ga unigram.
class fallback_ngram:
    def __init__(self, models):
        self.models = models


    def autocorrect(self, text:str):
        new_tokens = []

        # Hubungin kata yang satu dengan yang lain
        tokens = word_tokenize(text)
        ngram_tokens = list(ngrams(tokens, 3, pad_left=True, left_pad_symbol = "<s>"))
        # Bikin list of list biar bisa dimutate
        ngram_tokens = [list(l) for l in ngram_tokens]

        # Jalanin
        for t in ngram_tokens:
            found = False
            for model in self.models:
                t = t[-model.n_gram:]
                result = model.prob_grams(t)
                if len(result):
                    print(f"Corrected {t} to: ", end="")
                    t[-1] = result.iloc[0]["word"][-1]
                    new_tokens.append(t[-1])
                    print(f"{t} ({model.n_gram}-gram)")
                    found = True
                    break
            if not found:
                new_tokens.append(t[-1])
        return TreebankWordDetokenizer().detokenize(new_tokens)
```

```python
[ ] combined_model = fallback_ngram([fourgram, trigram, bigram, unigram])
combined_res = combined_model.autocorrect(test)
```

```
Corrected ['of', 'gene'] to: ['of', 'one'] (2-gram)
Corrected ['a', 'proimising'] to: ['a', 'promising'] (2-gram)
Corrected ['expalined'] to: ['explained'] (1-gram)
Corrected ['regneration'] to: ['generation'] (1-gram)
Corrected ['tarhgting'] to: ['targeting'] (1-gram)
Corrected ['markeers'] to: ['markets'] (1-gram)
```

Link colab:

https://colab.research.google.com/drive/1crgFYiyz9s

Methods

Evaluation

- Untuk mengevaluasi model, kami mengevaluasi masing-masing dari 6 metode dengan hitungan sebagai berikut:

Correctly Spelled Words = no. of correctly spelled words / total number of words

- Kami menghitung correctly spelled words menggunakan Python. Hasilnya dalam tabel sebagai berikut:

	Original text	Unigram	Bigram	Trigram	Fourgram	1-4 gram*	1-2 gram*
Correctly spelled words (ratio)	420 / 519	460 / 519	440 / 519	426 / 519	421 / 519	467 / 519	467 / 519
Correctly spelled words (%)	80.92%	88.63%	84.77%	82.08%	81.11%	89.98%	89.98%

*combined model

Hitungan di Python:

```
Correctly spelt words for Original Text is 420 / 519. 0.80924
Correctly spelt words for Unigram is 460 / 519. 0.88631984585
Correctly spelt words for Bigram is 440 / 519. 0.847784200385
Correctly spelt words for Trigram is 426 / 519. 0.82080924855
Correctly spelt words for Fourgram is 421 / 519. 0.8111753371
Correctly spelt words for 1-4 Gram is 467 / 519. 0.8998073217
Correctly spelt words for 1-2 Gram is 467 / 519. 0.8998073217
```

Hasil evaluasi:

1. Combined model 1-4 gram dan 1-2 gram merupakan dua model terbaik dibanding model lainnya untuk mendeteksi dan mengoreksi typo.

2. Dalam merancang model n-gram typo detector, semakin tinggi nilai n belum tentu menyebabkan model bekerja lebih baik.

Conclusion

1

AKURASI MODEL

Semakin tinggi nilai n-gram belum tentu menyebabkan model bekerja lebih baik.

2

ARSITEKTUR & IMPLEMENTASI

Kecepatan spellchecking turun untuk n-gram yang lebih tinggi.

Ada baiknya memilih nilai n yang fixed agar dapat digunakan struktur data yang lebih cepat.

3

DATASET

Data training sebaiknya disesuaikan dengan penggunaan ATAU dikumpulkan lebih banyak untuk menghindari OOV.

Conclusion

Model	Runtime	Entries
Unigram	52s	63 K
Bigram	50s	441 K
Trigram	78s	930 K
Fourgram	110s	1 217 K
1-4 gram	228s	2 651 K
1-2 gram	67s	504 K

Kecepatan spellchecking

freq	word1	word2
2665	for	the
19	the	the
4	vs	vs
2	for	for
1	vs	the

word2	for	the	vs
word1			
for	2	2665	0
the	0	19	0
vs	0	1	4

Data dapat dibuat n-dimensional untuk mempercepat komputasi

Conclusion

```
renowned
   word  freq

renewed
            word  freq
6088  (renewed,)    67
```

```
Corrected ['renowned'] to: ['renewed']
```

"renowned" sebagai contoh OOV yang dianggap typo

THANK YOU