# PERBANDINGAN METODE PETER NORVIG, LSTM, DAN N-GRAM UNTUK *SPELL CORRECTION BAHASA INDONESIA*


Disusun Oleh:

N a m a : Anggasta Tirta Adi Kusuma
NIM : 19523126

**PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA**
**FAKULTAS TEKNOLOGI INDUSTRI**
**UNIVERSITAS ISLAM INDONESIA**
**2024**

**HALAMAN PENGESAHAN DOSEN PEMBIMBING**

# PERBANDINGAN METODE PETER NORVIG, LSTM, DAN N-GRAM UNTUK *SPELL CORRECTION BAHASA INDONESIA*

## TUGAS AKHIR

Disusun Oleh:

N a m a : Anggasta Tirta Adi Kusuma

NIM : 19523126

Yogyakarta, 09 Januari 2024

Pembimbing,

(Chanifah Indah Ratnasari, S.Kom., M.Kom.)

**HALAMAN PENGESAHAN DOSEN PENGUJI**

# PERBANDINGAN METODE PETER NORVIG, LSTM, DAN N-GRAM UNTUK *SPELL CORRECTION BAHASA INDONESIA*

## TUGAS AKHIR

Telah dipertahankan di depan sidang penguji sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer dari Program Studi Informatika – Program Sarjana di Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, 09 Januari 2024

Tim Penguji

Chanifah Indah Ratnasari, S.Kom., M.Kom

**Anggota 1**

Elyza Gustri Wahyuni, S.T., M.Cs.

**Anggota 2**

Zainudin Zukhri, S.T., M.IT.

Mengetahui,

Ketua Program Studi Informatika – Program Sarjana

Fakultas Teknologi Industri

Universitas Islam Indonesia

(Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D. )

**HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR**

Yang bertanda tangan di bawah ini:

Nama : Anggasta Tirta Adi Kusuma

NIM : 19523126

Tugas akhir dengan judul:

# PERBANDINGAN METODE PETER NORVIG, LSTM, DAN N-GRAM UNTUK *SPELL CORRECTION BAHASA INDONESIA*

Menyatakan bahwa seluruh komponen dan isi dalam tugas akhir ini adalah hasil karya saya sendiri. Apabila di kemudian hari terbukti ada beberapa bagian dari karya ini adalah bukan hasil karya sendiri, tugas akhir yang diajukan sebagai hasil karya sendiri ini siap ditarik kembali dan siap menanggung risiko dan konsekuensi apapun.

Demikian surat pernyataan ini dibuat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 09 Januari 2024


(Anggasta Tirta Adi Kusuma )

# HALAMAN PERSEMBAHAN

Pada kesempatan ini saya ingin mengucapkan rasa syukur keapada Allah SWT, penelitian ini saya persembahkan untuk Bapak dan Ibunda tercinta, Prapto Hartono dan Resnawati. Kepada orang tua saya yang selalu memberikan dukungan serta doa yang tanpa henti selalu dipanjatkan setiap harinya. Terima kasih kepada beliau lah sumber inspirasi dan kekuatan yang tak tergantikan. Terima kasih atas cinta, doa, dan dukungannya.

Saya juga ingin menyampaikan penghargaan dan terima kasih kepada ibu Chanifah Indah Ratnasari, S.Kom., M.Kom., sebagai dosen pembimbing saya. Ibu Chanifah telah dengan sabar dan penuh dedikasi membimbing serta memberikan dukungan sepanjang proses pengerjaan tugas akhir ini. Saya mengakui bahwa dalam perjalanan ini, mungkin ada kesalahan yang terjadi dari perkataan atau tindakan saya. Saya dengan tulus meminta maaf jika ada yang mungkin tidak disukai. Semoga Allah SWT memberikan balasan atas segala kebaikan dari kerja keras ibu.

## HALAMAN MOTO

“Jangan langsung menyerah! Semua orang itu memiliki kesempatan!”

(Asta)

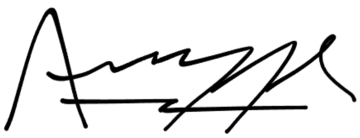
# KATA PENGANTAR

Assalamu‘alaikum wa rahmatullahi wa barakaatuh, Alhamdulillah, puji serta syukur kehadirat Allah SWT yang telah melimpahkan nikmat, rahmat, dan hidayah-Nya. Penulis berhasil menyelesaikan laporan tugas akhir berjudul Perbandingan Metode Peter Norvig, LSTM, dan N-gram Untuk *Spell Correction* Bahasa Indonesia.

Adapun penulisan skripsi ini dilakukan sebagai salah satu syarat yang harus dipenuhi dalam rangka menyelesaikan pendidikan tingkat Strata Satu (S-1) pada Program Studi Informatika, yang merupakan bagian dari Fakultas Teknologi Industri di Universitas Islam Indonesia. Penulis sangat menyadari bahwa pencapaian tugas akhir ini tidak dapat terlaksana tanpa kontribusi dan dukungan dari berbagai individu. Oleh karena itu, penulis ingin menyampaikan apresiasi yang mendalam kepada semua pihak yang telah memberikan bantuan dan dukungan dalam menyelesaikan skripsi ini terutama kepada:

1. Bapak Dr. Raden Teduh Dirgahayu, S.T., M.Sc. selaku Ketua Jurusan Informatika Fakultas Teknologi Industri Universitas Islam Indonesia
2. Bapak Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D. selaku Ketua Program Studi Informatika Program Sarjana Fakultas Teknologi Industri Universitas Islam Indonesia.
3. Kedua orang tua, Ayah Prapto Hartono dan Ibu Resnawati atas segala bentuk dukungan dan doa yang diberikan.
4. Ibu Chanifah Indah Ratnasari, S.Kom., M.Kom. selaku dosen pembimbing yang telah memberi semangat dan bimbingan dengan sabar dalam membantu menyelesaikan tugas akhir ini.
5. Diri sendiri yang telah berjuang sejauh dan sekuat ini dalam membuat penelitian ini.
6. Teman-teman dari seberapa imut club yang telah membantu serta memberikan dukungan dan semangat sehingga saya bisa menyelesaikan tugas akhir ini.
7. Buat sahabat saya Muthia Hana Ramadhan dan Bella Tiara Rahmah yang selalu memberikan dukungan moral, semangat, dan juga segala bentuk nasihat yang diberikan kepada saya.

Yogyakarta, 09 Januari 2024



(Anggasta Tirta Adi Kusuma )

# SARI

Kesalahan penulisan telah menjadi sebuah fenomena yang dianggap lumrah yang terjadi di era kemajuan teknologi saat ini, dimana kualitas akurasi data sering kali mengalami variasi yang signifikan. Oleh karena itu, diperlukan suatu bentuk pengolahan data yang lebih cermat dan hati-hati guna memastikan integritas serta ketepatan informasi yang disajikan. Keakuratan penulisan menjadi aspek yang sangat krusial, terutama ketika berbicara dalam konteks akademis di mana setiap kata dan frasa merepresentasikan arti atau maksud yang ingin disampaikan. Dalam upaya mereduksi kesalahan penulisan, terdapat salah satu implementasi *Natural Language Processing* (NLP) yaitu *spell correction*. Berbagai pendekatan dapat diambil dalam konteks ini, termasuk memanfaatkan metode yang telah dikembangkan oleh Peter Norvig, yang sering kali melibatkan penggunaan statistik dan probabilitas untuk menangani kesalahan penulisan. Selain itu, teknik *deep learning* seperti *Long Short-Term Memory* (LSTM) juga menjadi pilihan yang menjanjikan dalam mengenali pola kesalahan dan meresponnya dengan tepat. Namun, tidak hanya itu, konsep N-gram juga dapat diimplementasikan sebagai alat bantu dalam *spell correction* ini. Penggunaan fitur *spell correction* dalam penelitian ini bukan hanya sekadar untuk mengidentifikasi kesalahan penulisan semata, namun juga memberikan sugesti pembenaran kata yang salah tulis. Penelitian ini bertujuan untuk memberikan rekomendasi metode bagi peneliti di bidang NLP. Akurasi yang didapat untuk metode Peter Norvig ini menghasilkan 73% akurasi tertinggi dengan kecepatan waktu latih sebesar 31 detik, diikuti dengan N-gram 79% dengan kecepatan waktu latih sebesar 32 detik, dan LSTM (*Long Short Term Memory*) dengan akurasi 90,8% dengan kecepatan waktu latih sebesar 3 detik 451 milidetik

Kata kunci: *Spelling Correction*, Peter Norvig, LSTM, N-gram, *Spell Check*

# GLOSARIUM

| | |
|---|---|
| *Spell check* | Sebuah fitur mendeteksi kesalahan pengejaan dalam teks. |
| *Typographical error* | Kesalahan pengetikan atau cetak dalam suatu teks mencakup kesalahan pengejaan, pemilihan huruf yang salah, atau penyusunan huruf yang tidak benar. |
| *Preprocessing* | Tahap dalam pengolahan data yang bertujuan untuk membersihkan dan mempersiapkan data agar dapat diolah secara efektif oleh berbagai algoritma. |
| *Vanishing gradient* | Masalah pada LSTM dimana gradien yang digunakan selama proses pembelajaran arsitektur menjadi sangat kecil atau mendekati nol dan menyebabkan kesulitan dalam melatih model. |
| Korpus | Kumpulan teks atau data bahasa yang digunakan untuk penelitian linguistik, pemodelan bahasa, atau tugas pemrosesan bahasa alami. |
| *Dataset* | Kumpulan data yang diorganisir dan disusun untuk tujuan tertentu. Dataset memiliki variabel yang berhubungan untuk pembelajaran mesin atau analisis data. |

# DAFTAR ISI

# DAFTAR TABEL

# DAFTAR GAMBAR

# BAB I
# PENDAHULUAN

## 1.1 Latar Belakang

Pada tahun 2023, sebanyak 82% dari seluruh data internet terdiri dari teks, sebagaimana yang dilaporkan oleh Cisco Visual Networking Index. Hal ini menegaskan dominasi teks sebagai jenis data utama yang ada di internet. Data tekstual dapat ditemukan dalam berbagai bentuk, seperti artikel berita, blog, postingan media sosial, dan dokumen resmi. Untuk memberikan gambaran besarnya data tekstual di internet, per Januari 2022, Google memproses lebih dari 5,6 miliar pencarian setiap hari, di mana setiap pencarian mengandung teks dalam berbagai bahasa dan topik (Li et al., 2023). Karena informasi tersedia secara bebas di internet, pengumpulan dan pengelolaan data teks menjadi tugas yang lebih sulit. Kualitas, akurasi, dan signifikansi dari data ini bervariasi, oleh karena itu pemrosesan secara hati-hati diperlukan. Penulisan yang akurat dan bebas dari kesalahan eja menjadi hal yang sangat penting dalam konteks akademis (Fahma et al., 2018). Kesalahan ejaan dapat menyebabkan salah pemahaman dari maksud yang ingin disampaikan, keraguan mengenai kredibilitas tulisan, dan merusak tatanan tulisan dari karya tersebut. Oleh karena itu, tulisan dengan tanpa kesalahan eja menjadi sangat penting. Jelas bahwa sejumlah hal, seperti kesalahan tidak disengaja, masalah teknis, kelalaian atau kesalahan ketik, dan jarak dekat antar karakter pada keyboard, dapat mengakibatkan kesalahan ejaan pada penulisan pada teks (Jatminto & Nuryana, 2016).

*Spell check* merupakan alat dasar untuk melakukan *spell correction* (Gipayana, 2017). *Spell correction* menjadi salah satu solusi perkembangan teknologi yang dapat dimanfaatkan untuk melakukan pemeriksaan *typhograpical error* yang terjadi dalam tulisan atau dokumen yang dibuat. Ini dapat mengurangi kemungkinan kesalahan ejaan atau ketik dalam komunikasi, pencarian informasi, dan pembuatan dokumen (Mutammimah et al., 2017). Selain melakukan pemeriksaan kata, *spell correction* juga memiliki fitur ekstra yaitu *word suggestion*. Fitur ini bertujuan untuk membantu pengguna dengan memberikan sugesti daftar kata yang mirip dengan ejaan kata yang salah (Jatminto & Nuryana, 2016). *Typhograpical error* dibagi menjadi dua jenis yaitu, *real word error* dan *non-word error*. *Real word error* adalah kesalahan yang mengubah satu kata menjadi kata lain dengan makna yang sama atau berbeda, sedangkan *non-word error* adalah kesalahan yang mengubah suatu kata tertentu menjadi kata tanpa makna (Christanti et al., 2018).

Pada penelitian ini akan dilakukan perbandingan metode Peter Norvig, LSTM (*Long Short Term Memory*), dan N-gram dalam melakukan *spell correction* teks berbahasa Indonesia. Tujuan dari penelitian ini adalah untuk menguji metode dan membandingkan metode yang memiliki akurasi tertinggi serta perhitungan waktu dari setiap metode untuk *spell correction* teks Bahasa Indonesia. Metode Peter Norvig merupakan pendekatan yang dikembangkan oleh perusahaan mesin pencari dan menggabungkan algoritma khusus dengan memodifikasi kata yang salah, yang mana kata-kata salah tersebut akan melakukan pengecekan kembali untuk mencari apakah kata tersebut sesuai dengan kamus (Martin et al., 2021). Metode N-gram diterapkan dengan melakukan pertimbangan atau probabilitas kemungkinan kata-kata yang salah ditentukan dengan merujuk pada korpus data uji dalam data latih. Hasil penelitian yang dilakukan oleh Data latih berupa kalimat-kalimat yang dianalisis kata per kata beserta kata-kata sebelum dan sesudahnya (Simanjuntak et al., 2018). Penelitian yang dilakukan oleh Simanjuntak et al., (2018) membandingkan dua metode Peter Norvig dan N-gram, metode tersebut dapat memanfaatkan statistik probabilitas untuk menilai kemungkinan sebuah kata dalam korpus, dengan tujuan mencari kemunculan kata yang benar guna mengkoreksi ejaan kata yang salah. Hal ini dilakukan dengan mempertimbangkan probabilitas urutan kata-n yang muncul bersamaan dalam teks dan menghasilkan nilai akurasi sebesar 65,92%. Metode LSTM (*Long Short Term Memory*) memiliki kemampuan untuk memahami konteks kalimat dalam sebuah kalimat atau teks, dan memungkinkan model untuk beradaptasi dengan perubahan dalam penggunaan kata atau ejaan yang umum. Metode LSTM juga dapat digunakan untuk mendeteksi kesalahaan ejaan, memiliki sebuah jaringan saraf yang digunakan dalam proses deteksi untuk mengidentifikasi kata-kata yang salah dan lokasi kesalahannya. Menurut penelitian yang dilakukan Soisoonthorn et al., (2023) menghasilkan nilai akurasi sebesar 81,50% dan 72,18% untuk data latih dan data test.

Penelitian terdahulu sudah banyak membahas tentang perbandingan metode antara Peter Norvig dan N-gram, tetapi masih sedikit membandingkan dengan metode LSTM (*Long Short Term Memory*). Oleh karena itu, penelitian ini akan melakukan perbandingan antara Peter Norvig, N-gram, dan LSTM (*Long Short Term Memory*) pada *spelling correction* untuk mengetahui metode mana yang lebih baik dalam melakukan pengoreksian dan perbaikan ejaan kata.

## 1.2 Rumusan Masalah

Berdasarkan latar belakang tersebut dapat dirumuskan permasalahan penelitian, seperti di bawah ini:

a) Bagaimana membangun model untuk membuat *spell correction* Bahasa Indonesia dengan menggunakan metode Peter Norvig, LSTM, dan N-gram?
b) Dari ketiga metode berikut: Peter Norvig, LSTM, dan N-gram, metode manakah yang paling baik dalam melakukan *spell correction* untuk Bahasa Indonesia?

## 1.3 Batasan Masalah

Penelitian ini akan dibatasi oleh hal-hal berikut:

a) Bahasa yang akan diuji adalah Bahasa Indonesia dengan korpus Bahasa Indonesia hanya menggunakan label kesalahan ejaan *insertion*, *deletion*, dan *subtitution*.
b) Sistem *spelling correction* hanya dapat mendeteksi kesalahan kata pada dokumen dan tidak dapat memperbaiki kesalahan tanda baca.

## 1.4 Tujuan Penelitian

Adapun tujuan dari penelitian ini adalah:

a) Mengukur kinerja algoritma Peter Norvig, LSTM (*Long Short Term Memory*), dan N-gram dalam melakukan *spelling correction* pada dokumen yang menggunakan Bahasa Indonesia.
b) Membandingkan nilai akurasi terbaik antara metode Peter Norvig, LSTM (*Long Short Term Memory*), dan N-gram.

## 1.5 Manfaat Penelitian

Penelitian ini membandingkan metode Peter Norvig, LSTM (*Long Short Term Memory*), dan N-gram dalam melakukan *spelling* teks berbahasa Indonesia, sehingga penelitian ini bermanfaat dalam:

a) Mengetahui metode mana yang memiliki akurasi paling baik dalam melakukan *spelling correction* teks berbahasa Indonesia.
b) Memberi rekomendasi bagi peneliti-peneliti di bidang NLP dalam menentukan metode yang tepat dalam penelitiannya yang bersesuaian dengan penelitian ini.

## 1.6 Metodologi Penelitian

Langkah-langkah yang digunakan dalam penelitian ini untuk mencapai tujuan yang diinginkan adalah:

a) Studi pustaka

Studi pustaka dilakukan untuk mengumupulkan, meneliti, dan menganalisis penelitian terdahulu dan dasar teori yang berkaitan dengan penilitian ini dengan tujuan untuk memahami masalah penelitian yang akan dilakukan.

b) Pemilihan metode *spell correction*

Setelah melakukan studi pustaka, dilakukan evaluasi kelebihan dan kekurangan masing-masing metode. Tahapan ini menjadi suatu pertimbangan penting dalam menentukan pilihan metode terbaik untuk melakukan *spell correction*.

c) Pengumpulan dan persiapan data

Setelah melakukan analisis pada penelitian terdahulu dan penentuan metode *spell correction* yang akan digunakan, selanjutnya dilakukan pengumpulan data. Data yang digunakan pada penelitian ini didapatkan melalui website *Wortschatz Leipzig* sebagai penyedia korpus yang mana data dikumpulkan secara otomatis dari sumber-sumber publik yang dipilih dengan cermat. *Dataset* tersebut digunakan sebagai data latih pada model spell correction dari metode, Peter Norvig dan N-gram. Sedangkan pada model spell correction dengan LSTM, menggunakan dataset dari website kaggle yaitu, SPECIL (*Spell Error Corpus for Indonesian Language*).

d) Pembangunan model

Pada tahap ini dilakukan pembangunan model untuk masing-masing metode *spell correction* yang telah dipilih pada tahap sebelumnya.

e) Pengujian model

Model yang sudah dibangun akan melalui tahapan pengujian untuk menilai seberapa baik kinerja setiap metode untuk memperbaiki kesalahan penulisan.

f) Pengujian model

Kesimpulan yang diambil mencakup evaluasi kinerja metode berdasarkan model yang sudah dibangun, ketepatan, dan akurasi. Hasil ini memberikan gambaran jelas tentang sejauh mana model dapat memperbaiki kesalahan penulisan.

## 1.7 Sistematika Penelitian

Dalam pembuatan laporan tugas akhir ini, terdapat sistematika penulisan yang terdiri dari beberapa bab sebagai berikut:

BAB 1 PENDAHULUAN

Pada bab ini berisi pembahasan latar belakang masalah, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metodologi penelitian, dan sistematika penulisan.

BAB II KAJIAN LITERATUR

Pada bab ini berisi pembahasan mengenai teori-teori terkait dengan penelitian *spell correction* Bahasa Indonesia dan tinjauan terhadap penelitian yang pernah ada mengenai perbandingan beberapa metode *spell correction*.

BAB III METODOLOGI

Pada bab ini berisi pengumpulan data, analisis data, dan desain rancangan model *spell correction* yang akan dibangun pada penelitian ini.

BAB IV HASIL DAN PEMBAHASAN

Pada bab ini berisi pembahasan mengenai hasil dan uji sistem dari beberapa metode *spell correction* yang telah dikembangkan pada penelitian ini.

BAB V KESIMPULAN DAN SARAN

Pada bab ini berisi kesimpulan serta terhadap penelitian yang telah dilakukan.

# BAB II
# KAJIAN LITERATUR

## 2.1 *Natural Language Processing* (NLP)

Sebuah teknologi yang disebut pemrosesan bahasa alami (*Natural Language Processing*/NLP) memungkinkan mesin untuk mengenali dan berkomunikasi dengan manusia menggunakan berbagai bahasa (Walelign et al., 2021). NLP memiliki tujuan untuk mewakili dan memproses bahasa manusia menggunakan komputer. Ini memungkinkan komputer merespons pertanyaan dengan mempertimbangkan petunjuk kontekstual dengan cara yang mirip dengan manusia. Pengecekan ejaan, otomasi pengetikan, filter spam, pesan teks suara, serta asisten virtual seperti Alexa, Siri, dan yang lainnya merupakan adalah contoh umum dari penggunaan NLP (Selvaraj et al., 2022).

## 2.2 *Typhograpical Error*

Kesalahan ejaan merupakan kesalahan yang muncul saat mengetik teks dan dapat mengubah makna kata bahkan kalimat. Hal ini mencakup kesalahan yang disebabkan oleh masalah mekanis, kesalahan penulisan, slip tangan, atau kesalahan karena ketidaktahuan penulis seperti kesalahan ejaan. Kesalahan ejaan ini bervariasi dari kesalahan ketik biasa hingga kesalahan dalam struktur bahasa yang digunakan, atau bahkan mengubah makna dari kata tersebut. Jenis-jenis kesalahan ini dapat dibagi menjadi dua, yaitu kesalahan kata *real word* dan *non word*. *Non word* adalah kesalahan yang menyebabkan suatu kata menjadi tidak memiliki makna, sementara *real word error* adalah kesalahan di mana kata yang ditulis sebenarnya benar dan memiliki arti dalam kamus, tetapi tidak sesuai dengan konteks kalimat atau memiliki makna yang berbeda, dan bahkan mungkin terdapat kesalahan tata bahasa dalam kalimat tersebut (Fahma et al., 2018).

## 2.3 *Spell Correction*

Setelah melakukan *preprocessing* seluruh data akan diproses melalui program pengecek ejaan (*spelling checker*). Setelah dilakukan *spell check*, apabila suatu kata terdeteksi salah eja, maka dilakukan *spell correction*. Dalam *spell check* ketika sebuah kata terdapat atau ditemukan dalam sumber leksikal, maka kata itu dianggap benar. Sumber leksikal dalam hal ini adalah pusat data yang berisi teks, daftar kata, atau informasi bahasa lainnya. Keberadaan *spell*

*correction* secara otomatis mengintervensi untuk meningkatkan akurasi dan keterbacaan teks pada teks dengan kesalahan penulisan kata (Neto et al., 2020).

Sistem *spell correction* dapat mengklasifikasikan kesalahan ejaan ke dalam kelompok-kelompok berikut:

1. Kesalahan mengetik adalah jenis kesalahan yang paling dasar dan dapat berupa penambahan, pengurangan, penggantian, dan penukaran karakter dalam teks.
2. Kesalahan ejaan adalah kesalahan yang muncul karena salah tafsir fonetik kata. Salah ketik merupakan cara lain terjadinya kesalahan ejaan; namun ini tidak selalu hasil kebetulan, melainkan juga dapat terjadi akibat kebiasaan atau ketidaktahuan.
3. Masalah ketidakmampuan kosakata, yang biasanya terkait dengan penggunaan yang salah dari sufiks, afiks, dan prefiks.

## 2.4 Metode-Metode *Spell Correction*

Penelitian ini bertujuan untuk mendalami beberapa metode yang digunakan dalam koreksi ejaan (*spell correction*), sebuah aspek penting dalam pengolahan bahasa alami. Dengan memahami dan membandingkan berbagai pendekatan metode *spell correction*.

### 2.4.1 Metode Peter Norvig

Salah satu metode yang digunakan dalam *spell* correction dalam penelitian ini adalah Peter Norvig. Peter Norvig merupakan metode yang dapat melakukan perubahan jarak antara kata yang salah ejaan atau menggantikan kata yang salah dengan 2 huruf, serta melakukan beberapa penyuntingan yang diperlukan untuk mengganti 1 kata dengan yang lainnya. Metode Peter Norvig memiliki satu tugas yang diberikan kepada algoritma pemeriksaan ejaan adalah mengidentifikasi kandidat kata dengan membagi, menghapus, menukar, mengganti, dan menyisipkan huruf. Di sisi lain hasil koreksi kata sangat bergantung pada daftar kata dalam korpus. Bisa jadi terdapat kesalahan dalam perbaikan kata yang dipilih jika kata kandidat yang berdekatan dengan kesalahan ejaan tidak ditemukan dalam korpus (Simanjuntak et al., 2018).

Peter Norvig memiliki kemampuan untuk menghasilkan semua kemungkinan kata dengan berbagai operasi jarak edit, termasuk penyisipan, substitusi, transposisi, dan penghapusan, untuk kata-kata yang terdeteksi sebagai *typo*. Operasi ini diterapkan secara berurutan pada setiap huruf dalam kata *typo*. Setiap operasi menghasilkan kata yang berbeda dari kata awal dan kata-kata ini kemudian dicocokkan dengan kamus yang ada. Kata-kata kandidat yang cocok dengan kata dalam kamus akan dinilai, dan yang sesuai akan digunakan sebagai pengganti kata *typo*. Persamaan (2.1) berupaya mengidentifikasi koreksi c, dengan kata asli w,

meningkatkan kemungkinan bahwa c adalah koreksi yang dimaksudkan dari semua koreksi kandidat lainnya. Persamaan berusaha untuk menemukan koreksi $c$ dari semua kandidat koreksi yang mungkin yang memaksimalkan probabilitas bahwa $c$ adalah koreksi yang dituju dengan kata asli.

$$correction(w) = argmax_{c \in candidates} P(c)P(w|c) \quad (2.1)$$

$$correction(w) = argmax_{c \in candidates} \frac{P(c)P(w|c)}{P(w)} \quad (2.2)$$

Dikarenakan $P(w)$ memiliki nilai yang sama untuk setiap kemungkinan kandidat $c$, persamaan dapat dihapus dan disusun ulang sesuai dengan persamaan (2.2). Mekanisme pemilihan menggunakan $argmax$, di mana kandidat dengan probabilitas tertinggi dipilih. *Candidate Model* $c \in candidates$ menunjukkan kandidat kata $c$ dari kumpulan $candidates$. Sedangkan Model Bahasa (*Language Model*) $P(C)$ mencerminkan probabilitas munculnya kandidat $c$ pada suatu korpus dokumen. Sementara itu, Model Kesalahan (*Error Model*) $P(w|c)$ menunjukkan probabilitas bahwa kata $w$ adalah teks yang dimaksud pada kandidat $c$. Dengan demikian, proses pemilihan koreksi ejaan ini melibatkan penggunaan $argmax$ untuk memilih kandidat dengan probabilitas tertinggi (Fahrudin et al., 2021).

### 2.4.2 Metode N-gram

Metode *spell correction* N-gram merupakan model statistik yang digunakan dalam pemrosesan bahasa alami untuk memprediksi kemungkinan munculnya sebuah kata. Teks atau kalimat dibagi menjadi urutan N-gram, atau kata-kata berurutan. Secara inti, metode N-gram merupakan probabilitas suatu kata akan muncul berdasarkan urutan kata sebelumnya dalam suatu teks atau kalimat (Hardiyanti, 2021).

Penggunaan N-gram sangat luas digunakan dalam berbagai konteks, seperti memprediksi kata, koreksi ejaan, pengenalan suara, perbaikan kesalahan kata, dan pencarian string. Pendekatan N-gram melibatkan pengambilan potongan kata dengan sejumlah karakter tertentu dalam sebuah kalimat. Implementasi metode N-gram menggunakan potongan karakter atau kata dengan panjang n. Nilai n menentukan klasifikasi N-gram, yaitu n = 1 menggambarkan unigram contohnya: “perubahan“, “iklim“, “dunia“. Bigram menggambarkan n = 2 contohnya: “perubahan iklim“, “iklim dunia“. Terakhir untuk trigram menggambarkan n = 3, dengan

contoh: “perubahan iklim dunia“. Dalam proses identifikasi kesamaan pada proses N-gram, N-gram memiliki nilai akurasi tertinggi di semua tingkat kata-gram. N-gram digunakan untuk menentukan kemungkinan kata yang benar dalam kegiatan koreksi ejaan. Persamaan umum untuk kemungkinan suatu N-gram dapat di lihat pada formula (2.3):

$$P(W_n|W_{n-N+1}^{n-1}) = \frac{c(w_{n-N+1}^{n-1}W_n)}{c(w_{n-N+1}^{n-1})} \quad (2.3)$$

Diketahui:

P = N-gram probabilitas

W = *word*

n = Index

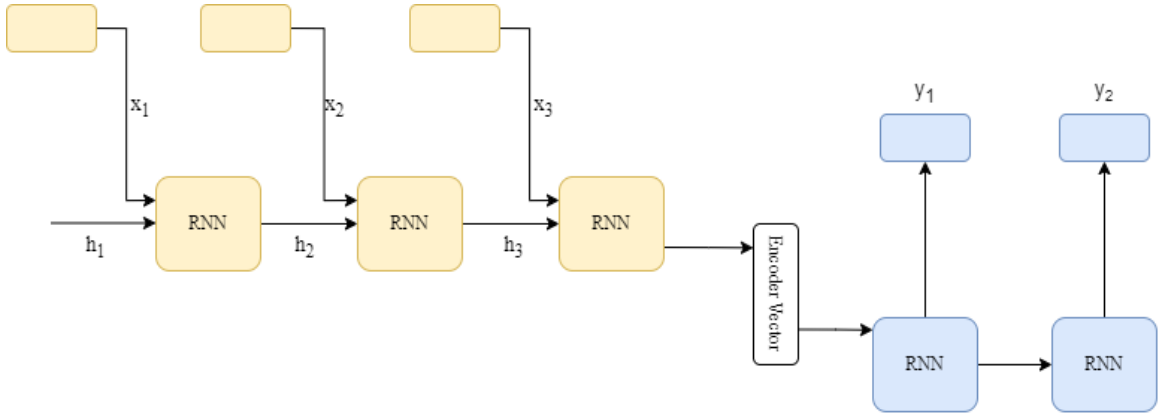
c = *Word frequency* (frekuensi kata)

Cara kerja *spell correction* menggunakan metode ini adalah sistem membaca kalimat sebagai masukan, membaginya menjadi potongan N-gram, memperbaiki masing-masing secara terpisah, dan kemudian menggunakan aturan probabilitas untuk menggabungkan potongan-potongan tersebut kembali menjadi satu kalimat (Christanti et al., 2018).

### 2.4.3 Metode LSTM (*Long Short Term Memory*)

Penelitian ini menggunakan model *sequence-to-sequence* berbasis LSTM. Paradigma *sequence-to-sequence* terdiri dari encoder dan decoder. Encoder memproses masukan dan membuat vektor atau konteks, yang kemudian dikirimkan ke decoder untuk menghasilkan keluaran. Sebagian besar dari urutan masukan hilang setelah memproses seluruh masukan, yang merupakan kelemahan dari arsitektur *sequence-to-sequence*. *Spell correction* dengan LSTM melibatkan dua proses, yaitu identifikasi dan perbaikan kesalahan. Dalam proses deteksi digunakan jaringan saraf dengan lapisan LSTM untuk mengidentifikasi kata-kata yang salah dan lokasinya. Berdasarkan prosedur perbaikan kesalahan menghasilkan kata-kata potensial untuk kata-kata yang salah eja.

Tugas bagian yang ditunjukkan dalam Gambar 2.1, encoder adalah mengubah kalimat masukan (sebuah urutan token) menjadi vektor fitur. Setiap elemen dalam informasi urutan masukan direpresentasikan oleh fitur-fitur tersembunyi ini. Untuk mengantisipasi kata atau karakter berikutnya, bagian decoder menggunakan keluaran dari encoder sebagai keadaan tersembunyi awal, kata atau karakter yang sedang dibaca, dan keadaan tersembunyinya sendiri.




Gambar 2.1 Arsitektur untuk Encoder dan Decoder

### 2.4.4 Metode Levenshtein *Distance*

Metode Levenshtein *Distance* merupakan metode pendekatan Metrik, yang pertama kali diperkenalkan oleh Levenshtein. Metode ini menghitung jarak edit untuk menentukan seberapa mirip dua kata satu sama lain. Jumlah minimum prosedur pengeditan dasar yang diperlukan untuk mengubah sebuah kata yang salah eja menjadi kata dalam kamus dikenal sebagai "jarak edit“. Metode untuk menentukan jarak Levenshtein antara dua *string*, $X = x_1x_2 \dots x_m$ dengan panjang m dan $Y = y_1y_2 \dots y_n$ dengan panjang n, adalah dengan menghitung jarak antara berbagai *sub-string* dari $X$ dan $Y$ secara iteratif dalam sebuah matriks berorde $m\ x\ n$ (Aouragh et al., 2015).

Metode Levenshtein *Distance* ini menggunakan pembobotan untuk menetapkan sejauh mana dua string atau kata serupa satu sama lain yang diperlukan untuk mengubah satu string menjadi *string* lainnya (similarity *score*) untuk setiap operasi pengeditan, termasuk substitusi, penghapusan, dan penambahan. Untuk menentukan Jarak Levenshtein antara dua kata, diperlukan rumus untuk menghitung persamaan matriks yang ditunjukkan pada persamaan (2.4).

$$M(i,j) = Min \begin{matrix} M((i,j-1)+1) \\ M\big((i-1,j)+1\big) \\ M((i-1,j-1)+d(q_i,l_j) \end{matrix}$$
$$where\ d(q_i,l_j = 0\ \ if\ q_i = l_j \qquad (2.4)$$
$$else\ d\big(q_i,l_j\big) = 1$$

Dengan menghitung secara berulang perbedaan antara kata yang salah eja $q_1, q_2, \dots q_i$ dan kata dalam kamus $l_1, l_2, \dots l_i$ sebuah fungsi $f(i,j)$ dilakukan perhitungan untuk semua huruf

dalam kata yang salah eja dan semua huruf dalam kamus. Setiap penambahan, penghapusan, atau substitusi diberi skor satu (Indah Ratnasariet al., 2017).

Tabel 2.1 Tabel perhitungan jarak Levenshtein

| | | M | O | B | I | L |
|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 |
| M | 1 | 0 | 1 | 2 | 3 | 4 |
| A | 2 | 1 | 1 | 2 | 3 | 4 |
| K | 3 | 2 | 2 | 2 | 3 | 4 |
| A | 4 | 3 | 3 | 3 | 3 | 4 |
| N | 5 | 4 | 4 | 4 | 4 | 4 |

Untuk menghitung jarak dengan Levenshtein *distance* dapat dilakukan menggunakan Tabel 2.1. Jarak Levenshtein antara dua kata diukur sebagai jumlah minimum operasi (penyisipan, penghapusan, atau penggantian) yang diperlukan untuk mengubah satu kata menjadi kata lainnya. Dalam kasus ini, nilai pada sel terakhir, yaitu di baris "N" dan kolom "L" (5,4), menunjukkan bahwa diperlukan lima operasi untuk mengubah "MOBIL" menjadi "MAKAN" (Yulianto et al., 2018).

### 2.4.5 Metode Damerau Levenshtein

Dalam proses mengubah satu *string* menjadi *string* lain, Damerau Levenshtein digunakan untuk meningkatkan akurasi Levenshtein. Damerau Levenshtein melibatkan penyisipan, penghapusan, dan substitusi dengan tambahan operasi transposisi antara dua karakter (Hamidah et al., 2020). Berikut adalah beberapa contoh bagaimana operasi Daemrau Levenshtein diterapkan:

1. Penyisipan (*insertion*) adalah karakter pada indeks tertentu agar sesuai dengan karakter huruf sumber dan tujuan.
2. Penghapusan (*deletion)* adalah proses yang melibatkan penghapusan karakter pada indeks tertentu sesuai dengan karakter huruf sumber dan target.

3. Substitusi *(subtitution)* adalah proses penggantian karakter pada indeks tertentu sesuai dengan karakter huruf sumber dan target.
4. Transposisi *(transposition)* adalah proses mencocokkan karakter huruf dengan menukar karakter pada indeks tertentu (Santoso et al., 2019).

## 2.5 Penelitian Terdahulu

Untuk mencari referensi dari penelitian sebelumnya, peneliti melakukan pencarian literatur melalui layanan seperti Google Scholar, Research Gate, Science Direct, dan Academia Edu. Literatur yang dipakai memiliki rentang tahun sekitar 6 tahun yaitu 2017-2023. Kata kunci yang digunakan untuk melakukan pencarian literatur seperti "*Spell checking*", *"spell correction*", "*Spell correction NLP"*, "*Spell checking Indonesian Language"*, *"Spell correction Indonesian language*", "Peter Norvig", "N-gram", "*Long Short Term Memory*", dan *Spell correction using methods"*. Pencarian literatur dilakukan dengan tujuan mengukur dan membandingkan tingkat akurasi dan waktu dalam melakukan pembenaran kata-kata yang salah dengan membandingkan beberapa. Hasil identifikasi ini akan menjadi rujukan dalam penelitian ini.

Aytan & Sakar, (2023) membuat model untuk *spell correction* pada Bahasa Turki. Model ini memiliki dua tahap utama. Tahap pertama yaitu kata masukan pertama-tama diproses oleh model yang dilatih dengan Bi-LSTM dengan pemisah kata BPE untuk tugas klasifikasi biner. Tahap kedua melibatkan pemberian sampel yang telah dicap sebagai positif, yaitu kata-kata yang salah eja ke dalam model pengurangan positif palsu, yang bertujuan untuk mengurangi jumlah prediksi positif palsu. Data uji pada penelitian ini melakukan *dataset* manual yang didapat dari Twitter dengan jumlah dataset yaitu 2.422 kata, 1.974 ejaan kata yang benar dan 448 ejaan yang salah. Penelitian ini menggunakan 80% *dataset* untuk pelatihan dan validasi dengan hasil akurasi keseluruhan sebesar 67%.

Pada penelitian yang dilakukan oleh Ahmadzade & Malekzadeh, (2021), model yang disarankan merupakan model urutan karakter di mana setiap pasangan kalimat yang tidak benar ejaannya dan yang benar dipelajari dan dimasukkan ke dalam model. *Attention mechanism*, *decoder*, dan *encoder* adalah tiga komponen dari model ini. Model *encoder* dan *decoder* dalam model yang disarankan terdiri dari lapisan LSTM dan lapisan *embedding*. Hasil pengujian untuk model menggunakan data kata aktual adalah sebagai berikut: 75% untuk edit 0, 90% untuk edit 1, dan 96% untuk edit 2.

Pada penelitian yang dilakukan oleh Etoori et al., (2018) membangun koreksi ejaan otomatis dalam bahasa indic yang menggunakan model *sequence-to-sequence*, penelitian ini menyelesaikan masalah koreksi ejaan untuk bahasa Hindi dengan memanfaatkan model bahasa tersembunyi sebagai *decoder* dan jaringan korektor terpisah sebagai *encoder*. Dengan menggunakan dataset Bahasa Hindi, akurasi sistem ini ditemukan sebesar 72,3%. Penurunan akurasi dari akurasi awal HINSPELL dapat didistribusikan pada peningkatan ukuran data pengujian dan penambahan frasa yang tidak umum digunakan. Sebagai hasilnya, SCMIL melaporkan akurasi sebesar 85,4%, melebihi kinerja HINSPELL.

Penelitian yang dilakukan oleh Chaabi & Ataa. (2022) bertujuan untuk melakukan *spelling correction* pada Bahasa Amazigh dengan menggunakan beberapa metode yaitu Peter Norvig, BK-Tree, SymSpell, Linspell, N-gram, dan Damerau Levenshtein. Metode tersebut memiliki sumber *dataset,* yaitu DGLAI (*Dictionnaire General de la Langue Amazighe Informatisé*) yang berjumlah 520.000 data kata yang akan digunakan untuk beberapa langkah. Penelitian ini mendapatkan nilai F-*measure* pada tugas deteksi kesalahan ejaan berkisar antara 86,62% hingga 98,74%.

Penelitian yang dilakukan oleh Martin et al. (2021) bertujuan menemukan akurasi yang tepat dari metode Peter Norvig dan N-gram untuk mendeteksi tipo pada sistem *spelling correction* Bahasa Indonesia. Data dikumpulkan secara daring dengan mengajukan pertanyaan melalui platform media sosial kepada 10 mahasiswa yang diminta untuk memberikan 5 kalimat yang mengandung kesalahan ketik sehingga didaptkan 50 dokumen dan 5 dokumen tambahan dari peneliti dengan jumlah total dokumen yaitu 55 dokumen. Dengan nilai akurasi yang didapat yaitu 69,09%.

Penelitian lain dilakukan oleh Fahrudin et al. (2022) yang menguji kebenaran aplikasi KEBI, sebuah aplikasi berbasis web dengan algoritma untuk mendeteksi dan memperbaiki ejaan Bahasa Indonesia pada karya ilmiah. Diperlukan suatu skenario uji untuk memastikan akurasi dalam mengidentifikasi dan memperbaiki masalah ejaan dalam publikasi artikel ilmiah berbahasa Indonesia. Setelah memasukkan 10 naskah artikel ilmiah dalam Bahasa Indonesia, KEBI telah menunjukkan kinerja deteksi yang relatif baik dalam koreksi ejaan berdasarkan kesalahan tanda baca. Dalam mengoreksi kesalahan ejaan dalam Bahasa Indonesia, KEBI 1.0 *Checker* menawarkan kemampuan untuk tanda baca seperti penyambung ungkapan antar kalimat, konjungsi yang bertentangan. Menurut evaluasi kinerja, KEBI memberikan hasil yang baik, mencapai akurasi 76,67%, presisi 100%, dan recall 76,67%. Untuk penelitian yang dilakukan pada tahun sebelumnya Menurut temuan eksperimen, KEBI 1.0 Checker tampil

terbaik dalam hal akurasi, mencapai 100% ketika memperbaiki kata-kata non-standar dan hanya 55,52% dalam memperbaiki kesalahan ejaan.

Penelitian lain dilakukan oleh Purbaya et al. (2023) yang menganalisis sentimen ulasan pembelian *chip* menggunakan eksperimen perbandingan dengan mesin vektor dukungan kernel. Dengan pendekatan pengambilan data, ulasan pembelian chip di Shopee *Marketplace* menjadi sumber *dataset* yang digunakan dalam penelitian ini, dengan 84,31% dari 7.757 ulasan diklasifikasikan sebagai positif. Poin utama dari penelitian ini adalah teknik SVM, yang efektif saat dimodelkan dengan kernel linear. Dengan nilai kinerja akurasi sebesar 88,4%, metode Unigram memberikan hasil yang melampaui metode Bigram dan Trigram dalam ekstraksi fitur menggunakan metode TF-IDF dan model N-Gram.

Penelitian yang dilakukan oleh Ramadhan & Lhaksmana (2020), penelitian ini memiliki sistem pencarian hadis yang dihasilkan tanpa otentikasi dievaluasi dengan menggunakan data dari jaringan sosial. Setiap data dimasukkan sebagai pertanyaan dan dua penilaian dilakukan, satu dengan penerapan koreksi ejaan dan yang lainnya tanpa. Dibandingkan dengan tidak menggunakan *autocorrect*, yang menghasilkan mean *average precision* sebesar 73% dan *recall* rata-rata sebesar 80%, penggunaan modul *autocorrect* dalam penelitian ini dapat mengidentifikasi kesalahan ejaan dan memberikan saran untuk koreksi, menghasilkan *mean average precision* sebesar 81% dan *recall* rata-rata sebesar 89%.

Tabel 2.2 Studi literatur penelitian bahasa asing

| **Peneliti** | **Judul** | **Dataset** | **Jumlah Data** | **Metode** | **Akurasi** |
|---|---|---|---|---|---|
| (Aytan & Sakar, 2023) | *Deep learning-based Turkish spelling error detection with a multi-class false positive reduction model* | Kamus dari TDK (*Turkish Language Instutition*) | 2.422 kata | LSTM (*Long Short Term Memory*) | 67% |
| (Ahmadzade & Malekzadeh, 2021) | *Spell Correction for Azerbaijani Language using Deep Neural Networks* | Kamus Azerbaijani | 12.000 | LSTM (*Long Short Term Memory*) | 75%<br>90%<br>96% |
| (Etoori et al., 2018) | *Automatic Spelling Correction for Resource-Scare Languages using Deep Learning* | Shabdanjali kamus | 32.952 kata | LSTM (*Long Short Term Memory*) | 72,3%<br>85,4% |
| | | | | Peter Norvig | |

| | | | | | |
|---|---|---|---|---|---|
| (Chaabi & Ataa, 2022) | *Amazigh spell checker using Damerau-Levenshtein algorithm and N-gram* | DGLAI (*Dictionnaire General de la Langue Amazighe Informatise*) | 520.000 kata | N-gram<br>Bk-Tree<br>SymSpell<br>Linspell<br>Damerau Levenshtein | 86,62% - 98,74% |

Tabel 2.3 Studi literatur penelitian Bahasa Indonesia

| **Peneliti** | **Judul** | **Dataset** | **Jumlah Data** | **Metode** | **Akurasi** |
|---|---|---|---|---|---|
| (Martin et al., 2021) | Penggunaan *spelling correction* dengan metode Peter Norvig dan N-gram | Survey *online* | 55 dokumen | Peter Norvig<br>N-gram | 69,09% |
| (Fahrudin et al., 2022) | A rule-based spelling checker for correcting punctuation errors in Indonesia text using KEBI 1.0 checker | *Scientific article document* | 30 dokumen | Peter Norvig | 76,67% |
| (Fahrudin et al., 2021) | KEBI 1.0: Indonesian Spelling Error Detection System for Scientific Papers using Dictionary Lookup and Peter Norvig Spelling Corrector | *Students paper assignment* | 10 dokumen | Peter Norvig | 55,52% |
| (Purbaya et al., 2023) | Implementation of N-gram methodology to analyze sentiment reviews for Indonesian chips purchases in Shopee E-marketplace | Produk pembelian *Marketplace* Shopee | 7.757 kata | N-gram<br>SVM (*Support Vector Machine*)<br>TF-IDF (*Term Frequent-Inverse Document Frequency*) | 88,4% |
| (Ramadhan & Lhaksmana, 2020) | Improving Document Retrieval with Spelling Correction for Weak and Fabricated Indonesian-Translated Hadith | Indonesian *social media forums*, dan *news portal* | 85 hadis | Symspell<br>Peter Norvig | 89% dan 80% |

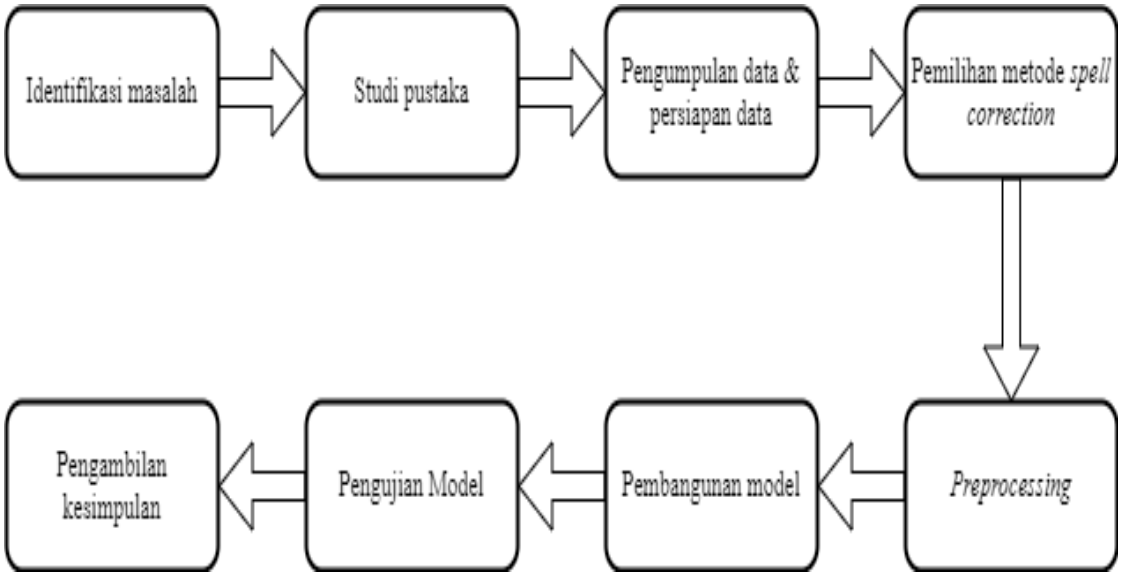
Berdasarkan berbagai penelitian dan jurnal sebelumnya, ditemukan bahwa banyak penelitian telah menggunakan berbagai macam metode. Namun, pada penelitian ini akan lebih

khusus membahas perbandingan metode Peter Norvig, N-gram, dan LSTM (*Long Short Term Memory)*. Sebagaimana disajikan dalam literatur, memberikan struktur dan kelancaran yang diperlukan dalam perbandingan *spell correction* ini.

# BAB III
# METODOLOGI PENELITIAN

## 3.1 Tahapan Penelitian

Pada penelitian ini dibangun model *spell correction* untuk teks Bahasa Indonesia menggunakan metode Peter Norvig, N-gram, dan LSTM (*Long Short Term Memory*). Metode Peter Norvig dan N-gram memanfaatkan probabilitas statistik untuk mengevaluasi kemungkinan kata dalam korpus, dengan tujuan mengidentifikasi kemunculan kata yang benar untuk mengoreksi ejaan yang salah. Ini dilakukan dengan mempertimbangkan probabilitas urutan kata-n yang muncul bersama dalam kata. Di sisi lain, penggunaan metode LSTM (*Long Short Term Memory*) memungkinkan pemahaman konteks kalimat dalam kata atau kalimat, memungkinkan model untuk menyesuaikan diri dengan perubahan dalam penggunaan kata atau ejaan yang umum. Penelitian ini bertujuan untuk mengembangkan sebuah model untuk pengujian dan perbaikan ejaan dalam Bahasa Indonesia serta menganalisis akurasi dari ketiga model tersebut. Gambar 3.1 menampilkan tahapan dari penelitian ini.




Gambar 3.1 Tahapan Penelitian

## 3.2 Studi Pustaka

Pada tahap ini, dilakukan studi terhadap penelitian terkait yang telah dilakukan sebelumnya, yang dapat dijadikan sebagai panduan untuk penelitian ini. Selain itu, dilakukan juga studi utnuk menganalisis metode yang sesuai untuk dilakukan perbandingan metode apa

yang memiliki keefektifan paling baik untuk dilakukan *spell correction*. Pada tahap ini menggunakan berbagai referensi, termasuk publikasi, jurnal ilmiah, dokumen digital, dan website.

Hasil pemaparan pada bab 2, pada Tabel 2.2 dalam penelitian yang membandingkan metode *spell correction* dengan menggunakan bahasa asing pada penelitian Chaabi & Ataa. (2022) menunjukkan akurasi yang cukup baik dihasilkan dengan penggunaan metode Peter Norvig, BK-Tree, Symspell, Linspell, N-gram, dan Damerau Levenshtein. Memiliki nilai akurasi yang didapat berkisar antara 86,62% hingga 98,74%. Nilai akurasi tersebut lebih unggul jika dibandingkan dengan nilai akurasi menggunakan metode LSTM (*Long Short Term Memory*). Karena penelitian tersebut menggunakan penggabungan beberapa metode.

Kemudian untuk penerapan perbandingan metode untuk Bahasa Indonesia ditunjukkan pada Tabel 2.3 pada penelitian Ramadhan & Lhaksmana. (2020) dengan menggunakan metode symspell dengan penggabungan metode Peter Norvig dengan menghasilkan akurasi 89% dan 80%. Pemilihan metode untuk penetlitian ini adalah Peter Norvig, N-gram, dan LSTM (*Long Short Term Memory*) ini didasarkan pada observasi bahwa ketiga metode tersebut telah menghasilkan hasil yang memuaskan dalam penelitian sebelumnya.

### 3.3 Pengumpulan Data

Penelitian ini menggunakan data yang didapatkan melalui salah satu website penyedia *dataset* yaitu *Wortschatz Leipzig*. Pada website ini terdapat *dataset* Bahasa Indonesia yang diambil pada tahun 2013 berupa bahan teks dari berbagai sumber seperti materi berita, teks web, wikipedia, dan lain-lain. Penelitian ini juga menggunakan data yang diambil dari *website* Kaggle yaitu, SPECIL (*Spell Error Corpus for Indonesian Language*) yang mana data ini akan digunakan sebagai data latih.

Selain pengumpulan data yang digunakan sebagai data latih, pada penelitian ini juga dibuat darta uji yang berisi kumpulan kata yang salah yang berjumlah 200 kata. Dalam data tersebut setiap data akan memiliki kesalahan kata seperti *insertion*, *deletion*, dan *subtitution*. Selanjutnya pada Tabel 3.4 menunjukkan beberapa contoh kata *insertion*. *Insertion* merupakan kesalahan yang terjadi ketika huruf atau suku kata disisipkan secara tidak sengaja di dalam kata.

Table 3.4 Contoh jenis kesalahan kata bertipe *insertion*

| No | Kata yang benar | Kata yang salah |
|---|---|---|
| 1. | **Bunyi** apa? | **Bunyti** apa? |
| 2. | Ceritakan kembali, mengapa **boni** kehilangan bolanya. | Ceritakan kembali, mengapa **boon**i kehilangan bolanya. |
| 3. | Amati bentuk huruf-huruf **di** atas. | Amati bentuk huruf-huruf **dio** atas. |
| 4. | Buatlah kartu **nama** kalian. | Buatlah kartu **namat** kalian. |
| 5. | Dengarkan guru yang **memberikan** petunjuk membuatnya. | Dengarkan guru yang **memberikano** petunjuk membuatnya. |
| 6. | Sekarang, perkenalkan diri kalian di depan **kelas**. | Sekarang, perkenalkan diri kalian di depan **kelats**. |
| 7. | Tunjukkan benda kesukaan kalian di kartu **itu**. | Tunjukkan benda kesukaan kalian di kartu **titu**. |
| 8. | Amati gambar **ini**. | Amati gambar **tini**. |
| 9. | Simaklah, lalu ikuti **guru** membacanya. | Simaklah, lalu ikuti **guoru** membacanya. |
| 10. | Aku mendengar **dengan** telingaku. | Aku mendengar **dengtan** telingaku. |

Tabel 3.5 menunjukkan contoh jenis kesalahan kata bertipe *deletion*. *Deletion* merupakan kesalahan yang terjadi ketika huruf atau suku kata dihilangkan secara tidak sengaja.

Tabel 3.5 Contoh jenis kesalahan kata bertipe *deletion*

| No | Kata yang benar | Kata yang salah |
|---|---|---|
| 1. | Tulis nama kalian pada **kartu**. | Tulis nama kalian ada **krtu**. |
| 2. | Lalu, gambar benda kesukaan kalian di **baliknya**. | Lalu, gambar benda kesukaan kalian di **balknya**. |
| 3. | Perkenalkan nama **kalian**. | Perkenalkan nama **kalan**. |
| 4. | Mengapa judul gambar itu pagi yang **sibuk**? | Mengapa judul gambar itu pagi yang **sibk**? |
| 5. | Tirukan juga cara **mereka** berjalan. | Tirukan juga cara **merea** berjalan. |
| 6. | Jawablah pertanyaan-pertanyaan yang dibacakan guru **berikut** ini. | Jawablah pertanyaan-pertanyaan yang dibacakan guru **berkut** ini. |

| | | |
|---|---|---|
| 7. | Mana yang lebih kalian sukai, siang **atau** malam hari? | Mana yang lebih kalian sukai, siang **ata** malam hari? |
| 8. | Amati gambar-gambar **ini**. | Amati gambar-gambar **ni.** |
| 9. | Aku mendengar **dengan** telingaku. | Aku mendengar **denan** telingaku. |
| 10. | Aku merasa **dengan** lidahku. | Aku merasa **denan** lidahku. |

Tabel 3.6 menunjukkan kesalahan kata bertipe *substitution*. Kesalahan ini terjadi karena salah satu huruf atau suku kata digantikan oleh huruf atau suku kata lainnya baik secara sengaja atau tidak sengaja.

Tabel 3.6 Contoh jenis kesalahan kata bertipe *subtitution*

| No | Kata yang benar | Kata yang salah |
|---|---|---|
| 1. | Hias sesukamu **dengan** menarik. | Hias sesukamu dengyn menarik. |
| 2. | Dokumen dan koleksi benda harus selalu **dirawat** dan dipelihara dengan baik. | Dokumen dan koleksi benda harus selalu **dirazat** dan dipelihara dengan baik. |
| 3. | Ktp dan sim biasanya dilaminating supaya **awet**. | Ktp dan sim biasanya dilaminating supaya **xwet**. |
| 4. | Pilihlah jawaban **yang** benar! | Pilihlah jawaban **wang** benar! |
| 5. | Jawablah pertanyaan berikut ini dengan singkat **dan** jelas! | Jawablah pertanyaan berikut ini dengan singkat **wan** jelas! |
| 6. | Apa yang kamu lakukan untuk merawat **dan** memelihara benda koleksi? | Apa yang kamu lakukan untuk merawat **wan** memelihara benda koleksi? |
| 7. | Bagaimana cara memelihara **koleksi** piala? | Bagaimana cara memelihara **koueksi** piala? |
| 8. | Kamu pasti **menyukai** cerita. | Kamu pasti **mevyukai** cerita. |
| 9. | Suatu hari roni **melihat** album foto di lemari. | Suatu hari roni **melzhat** album foto di lemari. |
| 10. | Kalau melihat foto, kita bisa mengenang **masa** lalu. | Kalau melihat foto, kita bisa mengenang **wasa** lalu. |

## 3.4 Pemilihan Metode *Spelling Correction*

Setelah melakukan studi pustaka dengan mempelajari beberapa referensi dan literatur yang sesuai, selanjutnya dianlisis untuk menentukan beberapa metode *spell correction* yang akan

dibandingkan dalam penelitian ini, yaitu: Peter Norvig, N-gram, dan LSTM. Untuk metode yang pertama yaitu Peter Norvig, menawarkan pendekatan statistik yang memanfaatkan korpus teks besar untuk menghitung probabilitas kemunculan kata dan memberikan saran koreksi berdasarkan alternatif yang paling mungkin. Tahapan ini melibatkan pembuatan kandidat koreksi dan peringkat mereka berdasarkan probabilitas. Metode yang kedua yaitu LSTM, melibatkan penggunaan jaringan saraf rekuren khusus yang dapat memahami ketergantungan kontekstual jangka panjang dalam urutan kata, menjadikannya efektif dalam menangani kata yang salah eja. Terakhir, model dengan metode N-gram akan diuraikan sebagai model bahasa statistik yang menganalisis probabilitas kemunculan urutan n kata atau karakter untuk memberikan koreksi berdasarkan probabilitas tersebut. Metode yang dipilih pada penelitian ini yaitu, Peter Norvig, N-gram dan LSTM. Peter Norvig dan N-gram dipilih sebagai pembanding untuk melakukan pengecekan kesalahan pada satu kata saja, sementara penggunaan LSTM (*Long Short Term Memory*) bertujuan untuk tambahan pembanding untuk melakukan pengecekan kesalahan eja yang lebih luas dengan menemukan kata yang benar dalam sebuah kalimat untuk membenarkan kesalahan eja pada sebuah kata.

Pada penelitian ini akan dilakukan evaluasi kinerja masing-masing metode untuk melakukan *spell correction* pada *dataset*, dengan mempertimbangkan faktor-faktor seperti akurasi, ejaan yang benar dan jenis-jenis kesalahan tertentu. Contoh kesalahan ejaan dan faktor penyebab kesalahan ditunjukkan dalam Tabel 3.7.

Tabel 3.7 Contoh kesalahan ejaan

| **Kata yang tidak benar** | **Potensi kata** | **Penyebab kesalahan** |
|---|---|---|
| Bserma | Bersama | Urutan huruf yang salah ‘e’,’r’,’s’ |
| Sleh | Salah | Kekurangan huruf ‘a’ dan menghapus huruf ‘e’ |
| | Saleh | Kekurangan huruf a |

## 3.5 *Preprocessing*

Sebelum data digunakan untuk proses lebih lanjut, data tersebut harus melewati serangkaian langkah atau prosedur. *Preprocessing* memiliki tujuan untuk membersihkan, mempersiapkan, dan mengorganisir data agar lebih mudah dipahami dan dimanfaatkan oleh

metode atau algoritma tertentu. Berikut adalah serangkaian *preprocessing* yang akan diakukan dalam penelitian ini adalah:

a. *Regular Expression*

   *Regular Expression* memiliki tujuan untuk membuat pola pencarian. *Regex* (*Regular Expression*) merupakan fungsi untuk mencocokkan, menemukan, dan mengekstrak teks dengan mengikuti pola tertentu berbasis string. *Regular Expression* akan mengekstrak karakter abjad "a" sampai "z" saja.

b. *Case Folding*

   *Case Folding* mengubah semua teks menjadi huruf kecil. Hal ini bertujuan untuk memudahkan pemrosesan dikarenakan format teks seragam, yaitu dalam huruf kecil semua.

### 3.6 Pembuatan Model

Setelah melakukan tahapan *preprocessing*, bagian ini akan memaparkan bagaimana ketiga metode yaitu, Peter Norvig, N-gram, dan LSTM (*Long Short Term Memory*) akan dibangun model *spell correction* yang dapat mendeteksi kesalahan eja pada teks Bahasa Indonesia.

### 3.7 Pengujian Model

Setelah model *spell correction* dari ketiga metode dibangun, selanjutnya dilakukan pengujian model. Dalam pengujian ini, dilakukan evaluasi efektivitas ketiga metode dalam memperbaiki ejaan kata-kata yang salah dalam teks. Metode Peter Norvig dan N-gram adalah pendekatan berbasis statistik yang mengandalkan model bahasa probabilistik untuk mengidentifikasi kemungkinan kata yang benar berdasarkan konteksnya. Sementara itu, metode LSTM adalah pendekatan berbasis jaringan saraf tiruan yang mampu memahami pola dalam data teks dan dapat digunakan untuk memprediksi kata yang benar dalam konteks yang lebih kompleks. Dengan menguji model *spell correction* dengan ketiga metode ini, penelitian ini bertujuan untuk memahami perbandingan kinerja dan kemampuan koreksi ejaan dari ketiga metode tersebut. Hasil dari penelitian ini akan memberikan wawasan yang berharga dalam pemilihan metode koreksi ejaan yang tepat dalam aplikasi pemrosesan bahasa Alami.

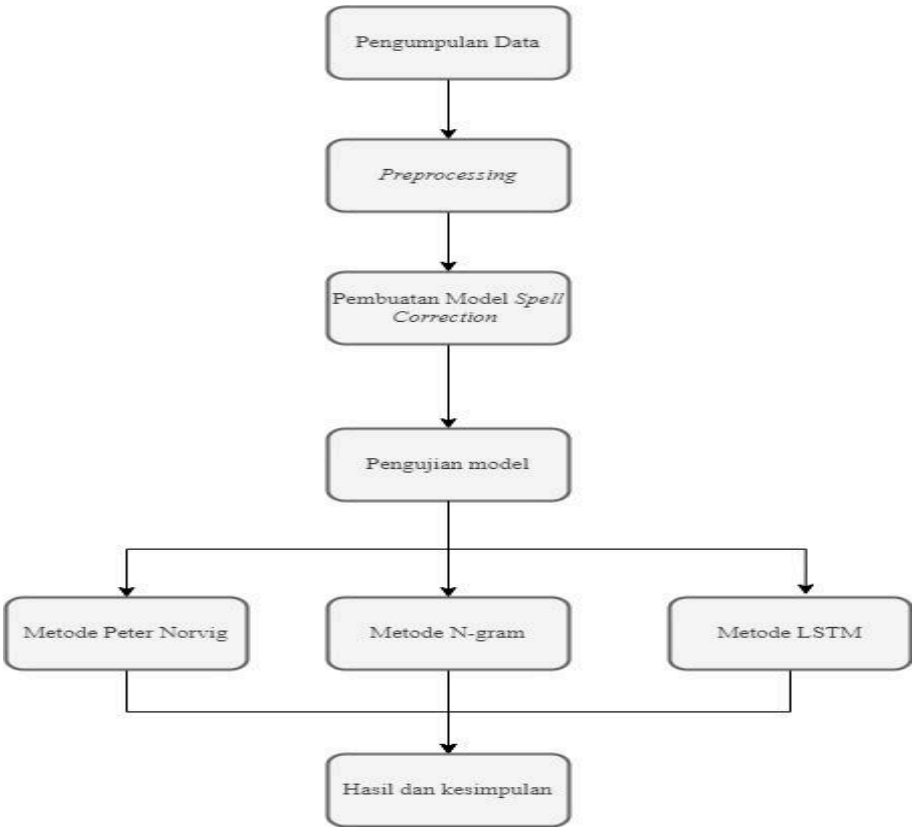
### 3.8 Pengambilan kesimpulan

Analisis hasil dari penelitian mencakup evaluasi akurasi koreksi ejaan dengan membandingkan metode yang memiliki hasil akurasi terbaik. Melalui interpretasi hasil, peneliti

berupaya mengidentifikasi keunggulan dan kelemahan masing-masing metode, serta memberikan wawasan yang mendalam terkait efektivitas koreksi ejaan dalam teks Bahasa Indonesia. Hasil dari perbandingan ini diharapkan dapat memberikan kontribusi berharga untuk pengembangan sistem koreksi ejaan yang lebih baik di masa depan.

# BAB IV
# IMPLEMENTASI DAN PENGUJIAN

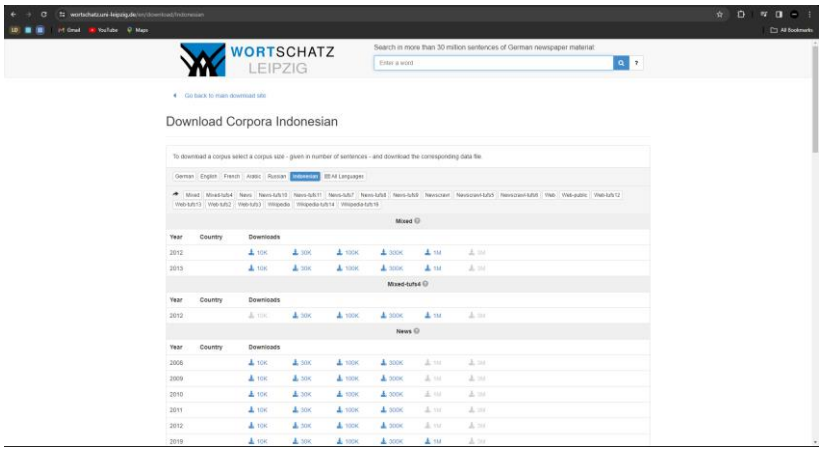
Di Bab 4 ini, *spell correction* teks Bahasa Indonesia dibangun dengan menggunakan Visual Studio Code versi 3.10 sebagai lingkungan pengembangan utama dan menggunakan ekstensi Jupyter Notebook. Setelah melakukan pengumpulan data, data tersebut akan digunakan untuk melakukan data uji pada metode Peter Norvig, N-gram, dan LSTM (*Long Short Term Memory*). Kemudian dilakukan tahapan *preprocessing* dengan mengklasifikasi setiap kata yang diuji dengan mengkategorikan hanya huruf saja, kemudian semua huruf akan diubah menjadi huruf kecil semua. Data yang telah melalui tahap *preprocessing*, langsung masuk ke tahap pembuatan model *spell correction* beserta tahapan pengujian model dari ketiga metode. Metode-metode tersebut setelah melakukan pengujian akan dibandingkan untuk menentukan perhitungan akurasi yang paling tinggi, serta melakukan perhitungan kompleksitas waktu untuk menentukan pertumbuhan waktu eksekusi dengan pertumbuhan ukuran input. Dengan kata lain, semakin besar teks atau kata yang harus dikoreksi, semakin lama waktu eksekusi yang dibutuhkan untuk ketiga metode.



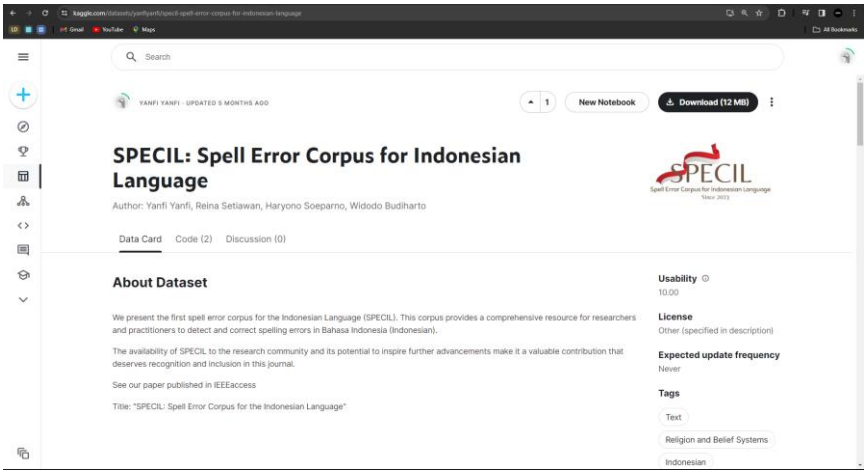

Gambar 4.1 Tahapan penelitian

## 4.1 Pengumpulan data

Proses pengumpulan *dataset* yang digunakan pada penelitian ini melalui website Wortshactz Leipzig dan *dataset* dari website kaggle SPECIL (*Spell Error Corpus for Indonesian Language*). Data ini sendiri dibedakan menjadi dua, pada metode Peter Norvig dan N-gram menggunakan data dari webiste *Wortshactz Leipzig* yang ditunjukkan dalam Gambar 4.2 sebagai data latih dengan jumlah kata benar dalam Bahasa Indonesia sebesar 10.000, dan untuk data uji dilakukan pengumpulan kata-kata yang memiliki kesalahan ejaan yang dikumpulkan secara manual dari kesalahan eja yang meliputi *insertion, deletion, dan substitution* sebanyak 200 kata di setiap kategori. Sedangkan LSTM (*Long Short Time Memory*) menggunakan data latih dan data tes yang sama dari website SPECIL (*Spell Error Corpus for Indonesian Language*) ditunjukkan melalui Gambar 4.3 dengan pengambilan tiga kategori kesalahan penulisan yaitu, *insertion*, *deletion*, dan *subtitution*, dengan banyak jumlah kata dari setiap kategori yaitu 21.500 kata. Penggunaan metode LSTM (*Long Short Term Memory*) menggunakan *dataset* yang berbeda ini dilakukan untuk melakukan uji coba pada kesalahan ejaan yang memiliki tipe data kalimat kesalahan eja, sedangkan penggunaan Peter Norvig dan N-gram hanya bisa memeriksa kesalahan eja dari kata atau huruf.




Gambar 4.2 Tampilan website *Wortshactz Leipzig*




Gambar 4.3 Tampilan website Kaggle SPECIL (*Spell Error Corpus for Indonesian Language*)

## 4.2 Implementasi *Preprocessing*

*Preprocessing* dilakukan melalui serangkain langkah yang masing-masing memiliki peran tertentu.

```
import re
def process(sent):
    sent=sent.lower()
    sent=re.sub(r'[^0-9a-zA-Z]',' ',sent)
    return sent
```

Gambar 4.4 Implementasi *preprocessing*

a. *Regular Expression*

Pada tahap ini bertujuan untuk membersihkan teks dengan menghilangkan semua karakter yang bukan merupakan huruf atau angka. Setiap karakter yang tidak termasuk dalam kategori huruf atau angka akan dihapus dari teks. Untuk implementasi *regular expression* dapat dilihat dari fungsi *regex* yang ada pada Gambar 4.4.

b. *Case Folding*

Pada tahap ini bertujuan untuk mengubah seluruh karakter dalam teks menjadi huruf kecil. Dengan melakukan *lowercasing*, perbedaan huruf besar dan kecil diabaikan, sehingga kata-kata yang sama dapat dianggap identik. Implementasi pada tahap *case folding* dapat dilihat pada Gambar 4.4 baris ke-3.

## 4.3 Pembuatan Model *Spell Correction*

### 4.3.1 Metode Peter Norvig

Pada tahap ini dilakukan pembuatan model *spell correction* dengan menggunakan metode Peter Norvig. Konsep kunci dari metode ini adalah probabilistik dan pemahaman statistika dalam pemrosesan bahasa alami. Sebagai langkah awal, Norvig sering mengusulkan pendekatan yang mengintegrasikan model probabilistik untuk mengatasi masalah seperti tokenisasi, pemodelan bahasa, atau penentuan kata yang paling mungkin dalam suatu konteks.

```
import re
from collections import Counter
from nltk.tokenize import word_tokenize
def preprocess_text(text):
   text = text.lower()
    tokens = word_tokenize(text)

    return token
def words(text):
    return preprocess_text(text)

WORDS=Counter(words(open(wortschatz.leipzig.txt', encoding="utf8").read()))
```

Gambar 4.5 Pembentukan *language model*

Kode yang disajikan pada Gambar 4.5 merupakan bagian dari program Python yang berfungsi untuk pemrosesan teks dan analisis frekuensi kata. Program ini memulai dengan mengimpor beberapa modul penting: modul "`re`" untuk ekspresi reguler, yang biasanya digunakan dalam pencarian dan manipulasi string berbasis pola; "`Counter`" dari modul "`collections`", yang digunakan untuk menghitung jumlah kemunculan setiap elemen unik dalam sebuah koleksi; dan "`word_tokenize`" dari NLTK, sebuah pustaka populer dalam pemrosesan bahasa alami yang menyediakan fungsi untuk membagi teks menjadi kata-kata atau token.

Dalam program ini, fungsi "`preprocess_text`" dikembangkan untuk mengonversi teks masukan menjadi huruf kecil dan kemudian memecahnya menjadi token menggunakan "`word_tokenize`". Proses ini memastikan bahwa analisis frekuensi tidak terganggu oleh perbedaan kapitalisasi. Fungsi kedua "`words`", pada dasarnya adalah fungsi pembantu yang langsung memanggil "`preprocess_text`" dan mengembalikan hasilnya.

Inti dari program ini terletak pada pembuatan variabel "`WORDS`", yang diinisialisasi dengan hasil dari "`Counter`". "`Counter`" ini diisi dengan token kata yang telah diproses, yang diperoleh dengan membaca dan memproses teks dari file yang ditentukan, dalam hal ini

sebuah file "`.txt`" yang berisi *dataset*. *Dataset* ini dibaca dengan "`encoding UTF-8`" untuk menangani karakter internasional dengan benar. Akhirnya, "`Counter`" menghasilkan kamus frekuensi kata, yang menyediakan jumlah kemunculan masing-masing kata dalam *dataset* tersebut.

```
def edits1(word):
    letters    = 'abcdefghijklmnopqrstuvwxyz'
    splits     = [(word[:i], word[i:] for i in range(len(word) + 1)]
    deletes    = [L + R[1:]           for L, R in splits if R
    transposes = [L + R[1] + R[0] + R[2:] for L, R in splits if len(R)>1]
    replaces   = [L + c + R[1:]           for L, R in splits if R for c in
    letters]
    inserts    = [L + c + R               for L, R in splits for c in
    letters]
return set(deletes + transposes + replaces + inserts)
```

Gambar 4.6 Kandidat model

Fungsi `` `edits1(word)` `` pada Gambar 4.6 adalah implementasi dalam Python untuk menghasilkan semua kemungkinan kata yang dapat diperoleh dengan satu operasi pengeditan dari kata input `word`. Proses tersebut melibatkan pembagian kata `` `word` `` menjadi pasangan kata (*splits*), di mana setiap pasangan terdiri dari dua bagian yang terbentuk dari pemisahan pada setiap posisi karakter. Dari sini, berbagai operasi pengeditan dilakukan, termasuk penghapusan satu karakter (*deletes*), transposisi dua karakter berturut-turut (*transposes*), penggantian satu karakter dengan semua huruf alfabet kecil (*replaces*), dan penyisipan satu karakter dari alfabet kecil ke setiap posisi dalam kata (*inserts*). Hasil dari semua operasi ini digabungkan dalam satu set untuk menghindari duplikat, dan set tersebut kemudian dikembalikan sebagai *output* dari fungsi. Dengan pendekatan ini, fungsi `` `edits1` `` berguna dalam konteks pengolahan kata dan pencarian berbasis kata dengan mempertimbangkan variasi hasil dari satu operasi pengeditan. Kemudian pembentukan variasi hasil dari suatu operasi pengeditan dilakukan perhitungan probabilitas yang ditunjukkan pada Gambar 4.7.

### 4.3.2 Metode N-gram

N-gram digunakan untuk memprediksi elemen berikutnya dalam urutan berdasarkan elemen-elemen sebelumnya, dengan ide utama bahwa probabilitas kemunculan suatu kata tidak hanya tergantung pada kata itu sendiri, tetapi juga pada kata-kata yang mendahuluinya. Penggunaan kode pada N-gram dan Peter Norvig merupakan hasil kolaborasi dengan menggabungkan logika seperti Gambar 4.5, Gambar 4.6, dan Gambar 4.7.

```
def P(word, N=sum(WORDS.values())):

return WORDS[word] / N
```

Gambar 4.7 Pembentukan *language model* dengan mencari probabilitas

Pada Gambar 4.7 fungsi "P" yang didefinisikan dalam kode ini merupakan fondasi statistik untuk model bahasa yang dibangun atas *dataset* teks. Tujuan fungsi ini adalah untuk menetapkan probabilitas empiris pada setiap kata tertentu dalam konteks keseluruhan teks. Ini dilakukan dengan membagi frekuensi kemunculan kata yang diinginkan diakses melalui "Counter" Gambar 4.5 yang sebelumnya telah didefinisikan sebagai "WORDS" dengan total jumlah kata dalam *dataset*. Total ini dihitung sekali sebagai jumlah agregat dari semua nilai dalam "WORDS" dan diwakili oleh "N". Dengan memanfaatkan operasi pembagian ini, fungsi "P" memberikan nilai desimal yang mewakili peluang munculnya kata tertentu setiap kali kata diambil secara acak dari dataset. Melalui pendekatan ini, dapat diukur dengan seberapa sering kata-kata tertentu muncul relatif terhadap keseluruhan korpus teks, yang merupakan langkah penting dalam pemahaman distribusi kata dan aplikasi praktis dari model bahasa, seperti pemeriksaan ejaan otomatis dan fitur prediksi teks yang digunakan dalam banyak perangkat lunak komunikasi modern.

### 4.3.3 Metode LSTM (*Long Short Term Memory*)

Metode ini merupakan sebuah metode dalam bidang jaringan saraf rekuren yang secara khusus dirancang untuk mengatasi kendala "*vanishing gradient*" yang sering terjadi pada RNN konvensional. Dalam konteks pemrosesan bahasa alami atau tugas yang melibatkan urutan waktu, LSTM memainkan peran penting dalam memahami dan memodelkan dependensi jangka panjang dalam data.

```
char_set = list("abcdefghijklmnopqrstuvwxyz0123456789")
char2int = {char_set[x]: x for x in range(len(char_set))}
int2char = {char2int[x]: x for x in char_set}
print(char2int)
print(int2char)
```

Gambar 4.8 Indeks pemetaan antara karakter dan bilangan bulat

Kode pada Gambar 4.8 bertujuan untuk membuat dua pemetaan (*mapping*) antara karakter dan indeks numerik. Pertama, variabel 'char2int' adalah kamus yang memetakan setiap

karakter dalam ‘char_set’ (yang mencakup spasi, huruf kecil, dan angka 0 hingga 9) ke indeks numeriknya. Kedua, variabel ‘int2char’ adalah kamus yang memetakan indeks numerik ke karakter yang terkait. Dengan kata lain, indeks 0 di-*mapping* ke spasi (' '), indeks 1 di-*mapping* ke 'a', dan seterusnya. Penggunaan dua kamus ini dapat memudahkan konversi antara karakter dan indeks numerik, yang dapat berguna dalam konteks seperti pemrosesan teks atau pengolahan data numerik yang melibatkan representasi karakter.

```
num_samples = len(input_texts)
encoder_input_data = np.zeros( (num_samples , max_enc_len ,
len(char_set)),dtype='float32' )
decoder_input_data = np.zeros( (num_samples , max_dec_len ,
len(char_set)+2),dtype='float32' )
decoder_target_data = np.zeros( (num_samples , max_dec_len ,
len(char_set)+2),dtype='float32' )
```

Gambar 4.9 Persiapan struktur data

Selanjutnya setelah melakukan indeks pemetaan, pada Gambar 4.9 menunjukkan tahapan selanjutnya yaitu pembentukan struktur data untuk mengembangkan model. Pertama-tama, variabel “num_samples” ditetapkan untuk mewakili jumlah total teks input yang akan diproses. Kemudian, didefinisikan tiga matriks multidimensi menggunakan NumPy, yang semuanya diinisialisasi dengan nol. Matriks “encoder_input_data” dibuat dengan bentuk yang memuat jumlah sampel, panjang maksimum encoder, dan ukuran set karakter. Ini akan menyimpan representasi one-hot encoding dari setiap karakter dalam teks input yang akan dikodekan. Selanjutnya, “decoder_input_data” dan “decoder_target_data” juga dibentuk sebagai matriks dengan jumlah sampel dan panjang maksimum decoder, namun keduanya memiliki dimensi tambahan untuk menampung set karakter ditambah dua simbol tambahan, yang biasanya mencakup token awal dan akhir dari sebuah sekuens. “decoder_input_data” digunakan untuk menyimpan data input untuk *decoder* dalam model “seq2seq”, sementara “decoder_target_data” akan menyimpan data target yang *decode*r coba prediksi. Inisialisasi matriks berupa nol ini merupakan langkah awal dalam proses pembuatan model, dan nilainya akan diperbarui selama proses pelatihan. Matriks-matriks ini penting karena mereka menyediakan struktur yang dibutuhkan untuk menangkap informasi yang diperlukan dalam pemrosesan sekuensial karakter-karakter yang akan diolah oleh model pembelajaran mesin.

## 4.4 Pengujian model

### 4.4.1 Pengujian Peter Norvig

Kemudian untuk pengujian pertama yang dilakukan oleh Peter Norvig ditunjukkan pada Gambar 4.10. Fungsi "`correction`" bertujuan untuk mengidentifikasi bentuk yang paling tepat dari sebuah kata yang mungkin salah eja. Proses ini diawali dengan menghasilkan sejumlah kandidat untuk kata yang salah tersebut menggunakan fungsi "`candidates`". Fungsi "`candidates`", pada gilirannya, mencoba serangkaian strategi untuk menemukan bentuk yang benar: pertama, dengan memeriksa apakah kata tersebut sudah benar (dikenali dalam *dataset*) kedua, dengan menerapkan satu perubahan (seperti penambahan, pengurangan, atau penggantian huruf) pada kata tersebut; ketiga, dengan menerapkan dua perubahan. Dua perubahan dapat diketahui melalui kode "`edits2`" untuk menghasilkan kemungkinan perubahan yang memiliki dua jarak dari suatu kata. Jika semua upaya ini gagal, fungsi tersebut kembali pada kata asli. Di antara semua kandidat ini, "`correction`" kemudian memilih kata dengan probabilitas tertinggi berdasarkan model statistik (dinyatakan dengan fungsi "`P`"), yang mengukur kemungkinan munculnya setiap kata berdasarkan frekuensinya dalam *dataset*. Pendekatan ini memungkinkan sistem untuk secara efektif menyarankan koreksi yang paling mungkin berdasarkan data yang ada.

```
def correction(word): return max(candidates(word), key=P)

def candidates(word):
    return known([word]) or known(edits1(word)) or known(edits2(word)) or
[word]
```

Gambar 4.10 Sistem koreksi ejaan

Setelah menentukan sistem koreksi ejaan pada Gambar 4.10, kemudian dilakukan serangkaian *unit tests* yang ditunjukkan dalam Gambar 4.11. Fungsi "`unit_tests`" ini dirancang untuk secara sistematis menguji berbagai aspek dari algoritma koreksi ejaan. Setiap baris dalam fungsi memeriksa skenario koreksi ejaan tertentu menggunakan pernyataan "`assert`", yang memastikan bahwa *output* dari fungsi "`correction`" sesuai dengan harapan. Misalnya, ada tes untuk memverifikasi bahwa fungsi mampu menangani kesalahan ejaan yang memerlukan penyisipan, penggantian, atau penghapusan karakter, serta transposisi huruf. Beberapa tes khusus memeriksa skenario seperti kata yang sudah benar ('memang'), kata yang tidak dikenali oleh sistem ('kalkulatif'), dan bahkan menguji fungsi pemrosesan teks dasar seperti `words`, yang memecah teks menjadi daftar kata-kata yang disederhanakan. Selain itu,

tes ini juga memvalidasi integritas dari kamus frekuensi kata "WORDS", memeriksa jumlah total kata dan frekuensi spesifik beberapa kata umum. Dengan mengonfirmasi nilai-nilai ini, tes menunjukkan bahwa sistem mendasarkan koreksi ejaannya pada data yang akurat. Unit tests ini sangat penting untuk memastikan keandalan dan efektivitas sistem koreksi ejaan, memberikan jaminan bahwa fungsi-fungsi tersebut bekerja sesuai dengan harapan dalam berbagai kondisi yang berbeda.

```
def unit_tests():
    assert correction('panh') == 'panah'               # insert
    assert correction('przjuryt') == 'prajurit'        # replace 2
    assert correction('minjadi') == 'menjadi'          # replace
    assert correction('merupan') == 'merupakan'        # insert 2
    assert correction('teersebut') == 'tersebut'       # delete
    assert correction('naivgasi') =='navigasi'         # transpose
    assert correction('naivgasii') =='navigasi'        # transpose + delete
    assert correction('memang') == 'memang'            # known
    assert correction('kalkulatif') == 'kalkulatif'    # unknown
    assert words('This is a TEST.') == ['this', 'is', 'a', 'test']
    assert Counter(words('This is a test. 123; A TEST this is.')) == (
           Counter({'123': 1, 'a': 2, 'is': 2, 'test': 2, 'this': 2}))
    assert len(WORDS) == 34805
    assert sum(WORDS.values()) == 171674
    assert WORDS.most_common(10) == [('dan', 4644),
            ('yang', 4520),
            ('di', 3056),
            ('pada', 2277),
            ('dengan', 2012),
            ('dari', 1986),
            ('ini', 1835),
            ('untuk', 1739),
            ('dalam', 1417),
            ('tahun', 1162)]
    assert WORDS['dan'] == 4644
    assert P('kalkulatif') == 0
    assert 0.01 < P('dan') < 0.04
    return 'unit_tests pass'
```

Gambar 4.11 Pengujian algoritma koreksi ejaan

Peter Norvig menggunakan *dataset* yang dibikin secara manual dengan jumlah 600 kata dengan masing-masing dataset memiliki 200 kata yang memiliki kategori *insertion*, *deletion*, dan *subtitution*. Sampel untuk pengujian yang digunakan pada metode Peter Norvig ditunjukkan dalam Tabel 4.8.

Tabel 4.8 Sampel data uji untuk Peter Norvig

| No | Koreksi Ejaan | Variasi kesalahan ejaan |
|---|---|---|
| 1. | sekolah | Sekkolahh, sekolahh, sekkoolahh, seokolahh ,sekolahan |
| 2. | bermain | Brmain, bermyn, brmanin |
| 3. | kopi | kope kppi kapo |
| 4. | kredibel | Kredibl, kedibel, kredebil |

Setelah melakukan koreksi ejaan dilakukan pengujian pada Peter Norvig dengan kode program yang ditunjukkan pada Gambar 4.12. Fungsi pertama "`spelltest`", merupakan fungsi utama yang menjalankan tes. Fungsi ini menerima daftar pasangan kata yang benar dan salah, kemudian menerapkan fungsi koreksi ejaan pada setiap kata yang salah. Selama proses ini, ia menghitung jumlah kata yang dikoreksi dengan benar dan mengidentifikasi kata-kata yang tidak dikenali oleh sistem. Fungsi ini juga mengukur waktu yang dibutuhkan untuk menjalankan tes dan melaporkan efisiensi sistem dalam kata per detik. Jika mode verbose diaktifkan, fungsi ini mencetak detail setiap kesalahan dan koreksi yang dilakukan. Fungsi kedua yang ditunjukkan pada Gambar 4.13 yang berfungsi menilai dan mengoreksi efektivitas dari algoritma. "`Testset`", adalah fungsi pembantu yang memproses teks input untuk menghasilkan pasangan kata yang benar dan salah. Fungsi ini mengurai setiap baris teks yang berformat 'kata benar: kata salah1 kata salah2' menjadi daftar pasangan tuple, dengan setiap tuple berisi satu kata yang benar dan satu kata yang salah. Daftar pasangan ini kemudian digunakan oleh "`spelltest`" untuk menjalankan tes koreksi ejaan.

```
def spelltest(tests, verbose=False):
    import time
    start = time.perf_counter()
    good, unknown = 0, 0
    n = len(tests)
    for right, wrong in tests:
        w = correction(wrong)
        good += (w == right)
        if w != right:
            unknown += (right not in WORDS)
            if verbose:
                print('correction({}) => {} ({}); expected {} ({})'
                      .format(wrong, w, WORDS[w], right, WORDS[right]))
    dt = time.perf_counter() - start
    print('{:.0%} of {} correct ({:.0%} unknown) at {:.0f} words per second
'   .format(good / n, n, unknown / n, n / dt))
```

Gambar 4.12 Fungsi pengujian *spell test*

```
def Testset(lines):
    return [(right, wrong)
            for (right, wrongs) in (line.split(':') for line in lines)
            for wrong in wrongs.split()]

print(unit_tests())
spelltest(Testset(open('norvig_insertion.txt')), True)
```

Gambar 4.13 Fungsi pengujian *Testset*

```
correction(quarttett) => quarttett (0); expected quartet (0)
correction(gembiraa) => gembiraa (0); expected gembira (0)
correction(gembirah) => gembirah (0); expected gembira (0)
correction(gemira) => gempa (10); expected gembira (0)
correction(gembiraa) => gembiraa (0); expected gembira (0)
correction(gembirah) => gembirah (0); expected gembira (0)
correction(jangkik) => jangka (15); expected jangkrik (0)
correction(janggkrik) => janggkrik (0); expected jangkrik (0)
correction(janggkrik) => janggkrik (0); expected jangkrik (0)
correction(optiimis) => optimus (2); expected optimis (0)
correction(optiimis) => optimus (2); expected optimis (0)
correction(optimiss) => optimus (2); expected optimis (0)
correction(optimiss) => optimus (2); expected optimis (0)
73% of 958 correct (15% unknown) at 31 words per second
```

Gambar 4.14Hasil uji data Peter Norvig operasi *insertion*

Untuk kesalahan ejaan tipe '*insertion*' ditunjukkan pada Gambar 4.14, metode ini mencapai akurasi 73% yang diidentifikasi, dengan berhasil membenarkan sebanyak 958 kata. Namun, terdapat 15% kata yang tidak diketahui atau tidak dapat dibaca oleh sistem secara keseluruhan, dengan kecepatan pembenaran mencapai 31 kata per detik. Ini menunjukkan efisiensi yang cukup tinggi dalam mengatasi kesalahan dimana huruf tambahan disisipkan.

```
correction(eksprssif) => ekspresi (7); expected ekspresif (0)
correction(eksresif) => ekspresi (7); expected ekspresif (0)
correction(konseptul) => konseptor (1); expected konseptual (0)
correction(konseptal) => konseptor (1); expected konseptual (0)
correction(kolabrat) => labrat (1); expected kolaborasi (4)
correction(klabrat) => labrat (1); expected kolaborasi (4)
correction(kolarati) => kolarati (0); expected kolaborasi (4)
correction(transpransi) => transpransi (0); expected transparansi (0)
correction(transparnsi) => transparnsi (0); expected transparansi (0)
correction(korelsi) => koreksi (1); expected korelasi (1)
correction(kolabratif) => kolabratif (0); expected kolaborasi (4)
correction(klabratif) => klabratif (0); expected kolaborasi (4)
correction(kolaratif) => komparatif (1); expected kolaborasi (4)
67% of 550 correct (19% unknown) at 31 words per second
```

Gambar 4.15 Hasil uji data Peter Norvig operasi *deletion*

Pada kesalahan ejaan tipe '*deletion*' pada Gambar 4.15, akurasi turun menjadi 67% kata yang dapat diidentifikasi, dengan 550 kata berhasil dibenarkan. Namun terdapa sebanyak 19% kata tidak dapat dikenali oleh sistem secara keseluruhan, dengan waktu pembenaran yang sama dengan tipe 'insertion', yaitu 31 detik per kata. Ini mengindikasikan bahwa meskipun metode ini cukup efisien dalam waktu, ia menemui lebih banyak kesulitan dalam mengoreksi kesalahan ejaan yang melibatkan penghapusan huruf.

```
correction(sterillisasi) => serialisasi (1); expected sterilisasi (0)
correction(sterilysasi) => sterilysasi (0); expected sterilisasi (0)
correction(assimilassi) => assimilassi (0); expected asimilasi (0)
correction(assimilasy) => assimilasy (0); expected asimilasi (0)
correction(assimilasy) => assimilasy (0); expected asimilasi (0)
correction(klandesstinn) => klandesstinn (0); expected klandestin (0)
correction(klandestinn) => klandestinn (0); expected klandestin (0)
correction(clandestin) => clandestin (0); expected klandestin (0)
correction(perihellion) => perihellion (0); expected perihelion (0)
correction(periheliion) => periheliion (0); expected perihelion (0)
correction(periheleon) => periheleon (0); expected perihelion (0)
54% of 600 correct (36% unknown) at 13 words per second
```

Gambar 4.16 Hasil uji data Peter Norvig operasi *subtitution*

Opeasi ejaan '*subtitution*' yang ditunjukkan dalam Gambar 4.16. Memiliki akurasi 54% yang dapat diidentifikasi dengan berhasil membenarkan 600 kata. Namun, terdapat peningkatan signifikan dalam jumlah kata yang tidak dikenali oleh sistem, yaitu sebesar 36%,

dan waktu pembenaran per kata meningkat menjadi 13 detik. Hal ini menandakan bahwa kesalahan tipe 'substitution' merupakan tantangan yang lebih besar bagi metode ini

### 4.4.2 Pengujian N-gram

Dalam pengujian *spell correction* dengan metode N-gram, Peneliti mengamati bahwa metode ini sebagai alat yang efektif dalam pemrosesan teks dan pengoreksi ejaan. Melalui penggunaan N-gram, telah mengidentifikasi bahwa meskipun implementasi kode untuk metode ini dapat serupa seperti pada penjelasan kode Peter Norvig di atas, perbedaan signifikan muncul ketika mengevaluasi dan menganalisis data pengujian. Penekanan khusus adalah pada kasus di mana data pengujian hanya memiliki satu variasi kesalahan ejaan yang ditunjukkan dalam Tabel 4.9. Dengan penerapan metode N-gram ini dapat dilihat pada Gambar 4.17, Gambar 4.18, dan Gambar 4.19.

Tabel 4.9 Sampel data uji untuk N-gram

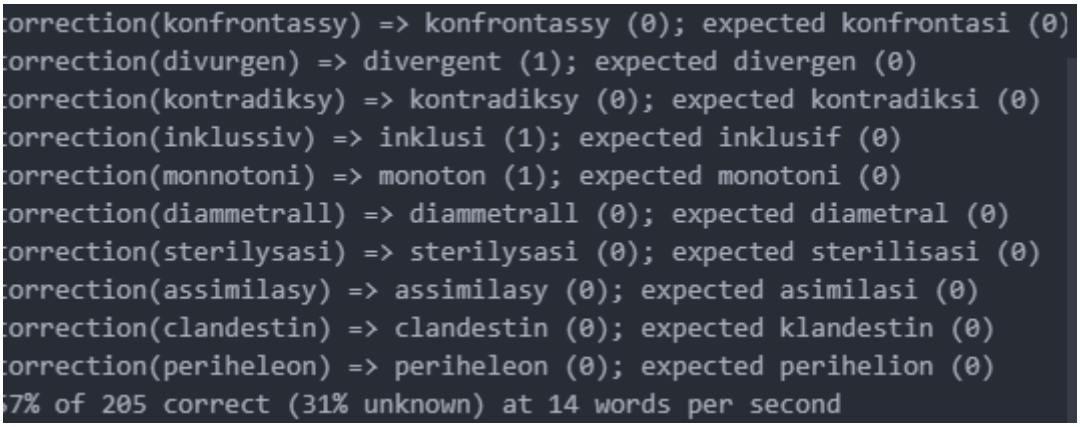
| No | Koreksi Ejaan | Variasi kesalahan ejaan |
|---|---|---|
| 1. | korelasi | korelasy |
| 2. | klandestin | clandestin |
| 3. | baju | bajuu |
| 4. | kemajuan | kemjuan |

```
correction(terass) => terasa (7); expected teras (3)
correction(urgenssi) => urgenssi (0); expected urgensi (0)
correction(vesspa) => tessa (1); expected vespa (0)
correction(xenofobii) => xenofobii (0); expected xenofobi (0)
correction(biisu) => biksu (5); expected bisu (3)
correction(dasterr) => master (4); expected daster (0)
correction(gulaii) => mulai (155); expected gulai (0)
correction(quarttet) => quarttet (0); expected quartet (0)
correction(gembiraa) => gembiraa (0); expected gembira (0)
correction(janggkrik) => janggkrik (0); expected jangkrik (0)
correction(optiimis) => optimus (2); expected optimis (0)
79% of 202 correct (16% unknown) at 32 words per second
```

Gambar 4.17 Hasil uji data N-gram operasi *insertion*

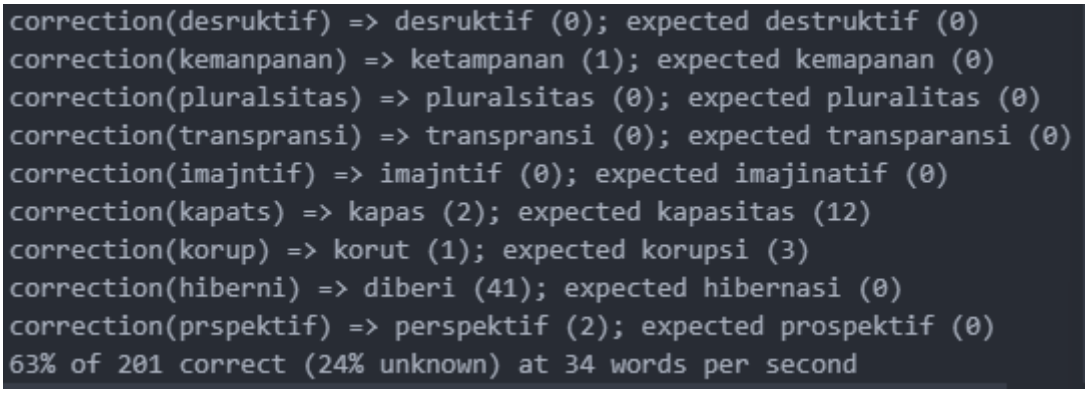
Untuk kesalahan ejaan tipe '*insertion*' pada Gambar 4.17, metode ini mencapai kata yang dapat diidentifikasi dengan menghasilkan akurasi 79%, dengan berhasil membenarkan 202 kata. Namun, terdapat 16% kata yang tidak dapat dibaca atau diketahui oleh sistem, dengan

kecepatan pembenaran mencapai 32 kata per detik. Ini menunjukkan efisiensi yang cukup tinggi dalam mengatasi kesalahan dimana huruf tambahan disisipkan.




Gambar 4.18 Hasil uji data N-gram operasi *subtitution*

Opeasi ejaan '*subtitution*' pada Gambar 4.18. Dapat mengidentifikasi secara keseluruhan kata dengan menghasilkan akurasi 57% dengan berhasil membenarkan 205 kata. Namun, terdapat peningkatan signifikan dalam jumlah kata yang tidak dikenali, yaitu sebesar 31%, dan waktu pembenaran per kata meningkat menjadi 14 detik. Hal ini menandakan bahwa kesalahan tipe '*substitution*' merupakan tantangan yang lebih besar bagi metode ini




Gambar 4.19 Hasil uji data N-gram operasi *deletion*

Pada kesalahan ejaan tipe '*deletion*' pada Gambar 4.19, akurasi turun menjadi 63% kata yang dapat diidentifikasi, dengan 201 kata berhasil dibenarkan. Di sini, sebanyak 24% kata tidak dapat dikenali oleh sistem, dengan waktu pembenaran, yaitu 34 detik per kata. Ini mengindikasikan bahwa meskipun metode ini cukup efisien dalam waktu, ia menemui lebih banyak kesulitan dalam mengoreksi kesalahan ejaan yang melibatkan penghapusan huruf.

### 4.4.3 Pengujian LSTM

Dalam pengujian LSTM (*Long Short Term Memory*) pada penelitian ini menggunakan jenis operasi *dataset* yang sama yaitu, *insertion*, *deletion,* dan *subtitution.*

```
batch_size = 128
epochs = 10
latent_dim = 256

num_enc_tokens = len(char_set)
num_dec_tokens = len(char_set) + 2
encoder_inputs = Input(shape=(None,num_enc_tokens))
encoder = LSTM(latent_dim,return_state=True)
encoder_outputs , state_h, state_c = encoder(encoder_inputs)
encoder_states = [state_h,state_c]
```

Gambar 4.20 Persiapan model *neural network*

Kode pada Gambar 4.20 memiliki beberapa variabel penting, seperti "`batch_size`"' yang menentukan ukuran batch data dalam proses pelatihan model, serta `epochs` yang menunjukkan berapa kali seluruh *dataset* akan digunakan selama pelatihan. Selain itu, kode memiliki "`latent_dim`" yang menggambarkan dimensi atau ukuran dari "latent space" yang digunakan oleh model untuk memahami fitur-fitur yang penting dalam data. Selain variabel tersebut, juga mendefinisikan "`num_enc_tokens`" untuk mengukur jumlah token atau karakter dalam input encoder, dan "`num_dec_tokens`" yang mencerminkan jumlah token dalam input decoder. Terakhir, mendefinisikan struktur input untuk encoder dengan menggunakan "`Input(shape=(None,num_enc_tokens)`", yang menciptakan tensor input dengan bentuk yang sesuai dengan panjang dan jumlah token yang ada dalam data input. Selanjutnya, pengujian mendefinisikan layer LSTM untuk encoder dengan ukuran "`latent_dim`" dan menginstruksikan agar layer ini mengembalikan state (`state_h` dan `state_c`) sebagai keluaran.

Kemudian hasil pengujian dilanjutkan pada Gambar 4.21. Pertama-tama, mendefinisikan input untuk decoder menggunakan "`decoder_inputs`", yang dapat menangani berbagai panjang sekuen. Selanjutnya, menentukan layer LSTM untuk *decoder* dengan ukuran "`latent_dim`". Layer ini menghasilkan urutan keluaran serta *state* yang digunakan dalam proses *decoding*, dan *state* ini diinisialisasi dengan informasi yang berasal dari *encoder* melalui "`initial_state`". Kemudian, mendefinisikan *layer dense* dengan aktivasi "`softmax`"

untuk menghasilkan output model dalam bentuk distribusi probabilitas token atau karakter dalam vocab decoder. Output dari layer LSTM dihubungkan ke *layer dense* ini, dan seluruh arsitektur model digambarkan dengan "`Model()`". Selanjutnya, model dikompilasi dengan menentukan "`optimizer`", fungsi kerugian, dan metrik untuk evaluasi performa model.

```
decoder_inputs = Input(shape=(None,num_dec_tokens))
decoder_lstm = LSTM(latent_dim,return_sequences=True,return_state=True)
decoder_ouputs,_,_     =     decoder_lstm(decoder_inputs,initial_state     =
encoder_states)

decoder_dense = Dense(num_dec_tokens, activation='softmax')
decoder_ouputs = decoder_dense(decoder_ouputs)

model = Model([encoder_inputs,decoder_inputs],decoder_ouputs)
model.compile(optimizer='rmsprop',loss='categorical_crossentropy',  metrics
= ['accuracy'])
early_stopping     =     EarlyStopping(monitor='val_loss',     patience=10,
restore_best_weights=True)
model.summary()

h=model.fit([encoder_input_data,decoder_input_data],decoder_target_data,
          epochs = epochs,
          batch_size = batch_size,
          validation_split = 0.2,
          callbacks=[early_stopping])
model.save('s2s.h5')
```

Gambar 4.21 Pengujian model LSTM

```
Epoch 1/10
7/7 [==============================] - 10s 644ms/step - loss: 1.1635 - accuracy: 0.0595 - val_loss: 0.9653 - val_accuracy: 0.0732
Epoch 2/10
7/7 [==============================] - 3s 489ms/step - loss: 0.9805 - accuracy: 0.0722 - val_loss: 0.9287 - val_accuracy: 0.0707
Epoch 3/10
7/7 [==============================] - 3s 493ms/step - loss: 0.9933 - accuracy: 0.0718 - val_loss: 0.9364 - val_accuracy: 0.0822
Epoch 4/10
7/7 [==============================] - 3s 479ms/step - loss: 0.9588 - accuracy: 0.0742 - val_loss: 0.9274 - val_accuracy: 0.0719
Epoch 5/10
7/7 [==============================] - 4s 579ms/step - loss: 0.9448 - accuracy: 0.0723 - val_loss: 0.9370 - val_accuracy: 0.0710
Epoch 6/10
7/7 [==============================] - 3s 473ms/step - loss: 0.9408 - accuracy: 0.0770 - val_loss: 0.9908 - val_accuracy: 0.0823
Epoch 7/10
7/7 [==============================] - 3s 474ms/step - loss: 0.9510 - accuracy: 0.0791 - val_loss: 0.9564 - val_accuracy: 0.0808
Epoch 8/10
7/7 [==============================] - 3s 501ms/step - loss: 0.9382 - accuracy: 0.0812 - val_loss: 0.9464 - val_accuracy: 0.0795
Epoch 9/10
7/7 [==============================] - 3s 490ms/step - loss: 0.9303 - accuracy: 0.0809 - val_loss: 0.8945 - val_accuracy: 0.0793
Epoch 10/10
7/7 [==============================] - 4s 537ms/step - loss: 0.9169 - accuracy: 0.0842 - val_loss: 0.8889 - val_accuracy: 0.0891
```

Gambar 4.22 Hasil pengujian LSTM operasi *insertion*

```
Epoch 1/10
7/7 [==============================] - 9s 611ms/step - loss: 1.1086 - accuracy: 0.0607 - val_loss: 1.0164 - val_accuracy: 0.0852
Epoch 2/10
7/7 [==============================] - 3s 444ms/step - loss: 1.0063 - accuracy: 0.0813 - val_loss: 0.9912 - val_accuracy: 0.0849
Epoch 3/10
7/7 [==============================] - 3s 429ms/step - loss: 0.9943 - accuracy: 0.0840 - val_loss: 0.9988 - val_accuracy: 0.0892
Epoch 4/10
7/7 [==============================] - 3s 447ms/step - loss: 0.9930 - accuracy: 0.0833 - val_loss: 1.0200 - val_accuracy: 0.0839
Epoch 5/10
7/7 [==============================] - 3s 434ms/step - loss: 0.9876 - accuracy: 0.0836 - val_loss: 0.9789 - val_accuracy: 0.0845
Epoch 6/10
7/7 [==============================] - 3s 456ms/step - loss: 0.9723 - accuracy: 0.0858 - val_loss: 0.9730 - val_accuracy: 0.0845
Epoch 7/10
7/7 [==============================] - 3s 430ms/step - loss: 0.9685 - accuracy: 0.0905 - val_loss: 0.9611 - val_accuracy: 0.0911
Epoch 8/10
7/7 [==============================] - 3s 438ms/step - loss: 0.9598 - accuracy: 0.0932 - val_loss: 0.9657 - val_accuracy: 0.0908
Epoch 9/10
7/7 [==============================] - 3s 443ms/step - loss: 0.9585 - accuracy: 0.0949 - val_loss: 0.9678 - val_accuracy: 0.0914
Epoch 10/10
7/7 [==============================] - 3s 451ms/step - loss: 0.9604 - accuracy: 0.0949 - val_loss: 0.9601 - val_accuracy: 0.0908
```

Gambar 4.23 Hasil pengujian LSTM operasi *deletion*

```
Epoch 1/10
5/5 [==============================] - 8s 846ms/step - loss: 1.0578 - accuracy: 0.0675 - val_loss: 0.9272 - val_accuracy: 0.0662
Epoch 2/10
5/5 [==============================] - 3s 518ms/step - loss: 0.8858 - accuracy: 0.0634 - val_loss: 0.8590 - val_accuracy: 0.0617
Epoch 3/10
5/5 [==============================] - 3s 509ms/step - loss: 0.8592 - accuracy: 0.0627 - val_loss: 0.8593 - val_accuracy: 0.0617
Epoch 4/10
5/5 [==============================] - 3s 527ms/step - loss: 0.8544 - accuracy: 0.0608 - val_loss: 0.8472 - val_accuracy: 0.0617
Epoch 5/10
5/5 [==============================] - 3s 579ms/step - loss: 0.8451 - accuracy: 0.0610 - val_loss: 0.8422 - val_accuracy: 0.0617
Epoch 6/10
5/5 [==============================] - 3s 541ms/step - loss: 0.8315 - accuracy: 0.0669 - val_loss: 0.8583 - val_accuracy: 0.0627
Epoch 7/10
5/5 [==============================] - 3s 515ms/step - loss: 0.8377 - accuracy: 0.0658 - val_loss: 0.8259 - val_accuracy: 0.0607
Epoch 8/10
5/5 [==============================] - 2s 511ms/step - loss: 0.8243 - accuracy: 0.0652 - val_loss: 0.8681 - val_accuracy: 0.0594
Epoch 9/10
5/5 [==============================] - 2s 496ms/step - loss: 0.8410 - accuracy: 0.0667 - val_loss: 0.8373 - val_accuracy: 0.0688
Epoch 10/10
5/5 [==============================] - 3s 531ms/step - loss: 0.8196 - accuracy: 0.0688 - val_loss: 0.8137 - val_accuracy: 0.0719
```

Gambar 4.24 Hasil pengujian LSTM operasi *subtitution*

Hasil dari penggunaan model LSTM dalam Gambar 4.20 menggunakan berbagai operasi pengujian menunjukkan sejumlah statistik yang dapat dianalisis. Dalam konteks pengujian *insertion* yang ditunjukkan dalam Gambar 4.22 model LSTM menghasilkan akurasi 89% dengan hasil perhitungan kecepatan waktu latih sebesar 4 detik 537 milidetik. Kemudian hasil pengujian untuk *deletion* mengalami kenaikan yang ditunjukkan dalam Gambar 4.23 menghasilkan 90,8% dengan hasil perhitungan kecepatan waktu latih sebesar 3 detik 451 milidetik, selanjutnya untuk operasi *subtitution* yang ditunjukkan dalam Gambar 4.24 mengalami penurunan akurasi yang cukup signifikan dengan menghasilkan akurasi yang paling rendah yaitu 71,9% dengan hasil perhitungan kecepatan waktu latih sebesar 3 detik 531 milidetik.

## 4.5 Kesimpulan dan Hasil

*Dataset* uji terdiri dari total 600 kata, dengan Peter Norvig dan N-gram menerapkan 200 kata operasi kata dari *insertion, deletion,* dan *subtitution*, berlandaskan korpus dari *Leipzig Corpora Collection*. Eksperimen menunjukkan bahwa Peter Norvig dan N-gram menghitung probabilitas kata yang diinput ke sistem dan menentukan persentase tertinggi sebagai kandidat paling mungkin. Namun, Norvig dan N-gram tidak selalu berhasil menemukan saran yang sesuai dari korpus, dan kadang gagal memberikan rekomendasi untuk kata-kata yang salah eja. LSTM unggul dalam memberikan koreksi ejaan yang akurat dan kontekstual hasil akhir dari setiap metode dapat dilihat pada Tabel 4.10.

Peter Norvig mencatat akurasi pada operasi *insertion* 73% dengan kecepatan 31 kata per detik, membenarkan 958 kata, namun 15% tidak dapat dikenali oleh sistem. Untuk *deletion* menghasilkan 67% dengan kecepatan yang sama seperti *insertion* yaitu, 31 kata per detik membenarkan 550 kata yang dapat dibaca, namun 19% kata tidak dapat dikenali oleh sistem selanjutnya untuk operasi *subtitution* mengalami penurunan sebesar 54% dengan kecepatan 14 kata per detik dan membenarkan 600 kata, dan menghasilkan 36% kata yang tidak dapat dikenali oleh sistem.

N-gram mencapai 79% akurasi pada operasi *insertion* dengan kecepatan 32 kata per detik, membenarkan 202 kata dan meninggalkan 16% kata yang tidak dapat diketahui oleh sistem. Operasi selanjutnya *deletion*, menghasilkan 63% dengan 201 kata berhasil untuk dibenarkan, menghasilkan 34 kata per detik. Selanjutnya sebanyak 24% kata tidak dapa dikenali oleh sistem. Operasi *subtitution* mengalami penurunan akurasi sebesar 57% dengan berhasil membenarkan 205 kata dan waktu pembenaran mencapai 13 kata per detik. Operasi *subtitution* mengalami peningkatan untuk kata yang tidak dapat dikenali oleh sistem sebesar 31%

Sementara itu, LSTM menunjukkan performa beragam dalam pengujian yang berbeda: pada pengujian *insertion*, model menghasilkan akurasi yaitu 89,1% dengan kecepatan waktu 4 detik 537 milidetik. Dalam pengujian *deletion*, model berhasil meningkatkan akurasi dan kecepatan waktu latih sebesar 90,8% dan 3 detik 451 milidetik. Hasil operasi uji *subtitution* menunjukkan LSTM mengalami penurunan akurasi yang cukup besar yaitu, 71,9% dengan kecepatan waktu latih 3 detik 531 milidetik.

Tabel 4.10 Hasil uji dan perbandingan metode-metode

| No | Metode | Akurasi | | | Waktu | | |
|---|---|---|---|---|---|---|---|
| | | ***Insertion*** | ***Deletion*** | ***Subtitution*** | ***Insertion*** | ***Deletion*** | ***Subtitution*** |
| 1. | Peter Norvig | 73% | 67% | 54% | 31 detik | 31 detik | 14 detik |
| 2. | N-gram | 79% | 63% | 57% | 32 detik | 34 detik | 13 detik |
| 3. | LSTM (*Long Short Term Memory*) | 89,1% | 90,8% | 71,9% | 4 detik 537 milidetik | 3 detik 451 milidetik | 3 detik 531 milidetik |

Penelitian ini menggunakan metode Peter Norvig dan N-gram yang menghasilkan akurasi antara 54%-79%. Perbedaan tersebut disebabkan oleh jumlah data uji yang digunakan, dimana metode LSTM (*Long Short Term Memory*) menggunakan jumlah data uji yang jauh lebih banyak. Metode Peter Norvig dan N-gram sama-sama menggunakan 600 total data, dengan masing-masing berjumlah 200 kata untuk operasi *insertion*, *deletion*, dan *subtitution*. Peter Norvig menggunakan jenis kesalahan ejaan yang lebih bervariasi, sementara N-gram menggunakan variasi yang lebih sedikit, terbatas pada unigram saja. Di sisi lain, LSTM menggunakan jumlah total data sebanyak 21.500 untuk setiap kategori insertion, deletion, dan substitution. Hal ini membuat metode LSTM (*Long Short Term Memory* ) menjadi lebih efisien dan fleksibel dalam memperbaiki kesalahan penulisan.

# BAB V
# KESIMPULAN DAN SARAN

Pada bab ini menjelaskan tentang kesimpulan yang diperoleh dari penelitian ini berdasarkan hasil yang diperoleh dari analisis proses, perbandingan akurasi, dan implementasi metode. Saran akan memberikan perbaikan yang dibutuhkan untuk penelitian yang akan datang.

## 5.1 Kesimpulan

Dari hasil penelitian yang telah dilakukan, dapat diambil beberapa kesimpulan sebagai berikut:

a. Model untuk membuat *spell correction* Bahasa Indonesia dengan menggunakan metode Peter Norvig, LSTM, dan N-gram dibangun dengan menggunakan bahasa Python. Untuk metode Peter Norvig dan N-gram menentukan probabilitas, sedangkan LSTM membangun dengan merancang arsitektur *encoder* dan *decoder*.
b. Metode yang paling baik dalam melakukan *spell correction* untuk Bahasa Indonesia adalah, Peter Norvig menghasilkan akurasi 73% untuk *insertion* dengan kecepatan proses waktu 31 detik *,* 67% untuk *deletion* dengan kecepatan proses waktu 31 detik*,* dan 54% untuk *subtitution* dengan kecepatan proses waktu 14 detik. N-gram menghasilkan akurasi 79% untuk *insertion* dengan kecepatan proses waktu 32 detik, 53% untuk *subtitution* dengan kecepatan proses waktu 34 detik, dan 67% *deletion* dengan kecepatan proses waktu 13 detik. Dan metode terakhir untuk LSTM menghasilkan 89,1% untuk operasi ejaan *insertion* dengan kecepatan waktu latih 4 detik 537 milidetik, 90,8% untuk operasi *deletion* dengan kecepatan waktu latih 3 detik 451 milidetik, dan 71,9% untuk operasi *subtitution* dengan kecepatan waktu latih 3 detik 531 milidetik. Pada hasil akhir memberikan operasi ejaan *insertion* memiliki hasil paling tinggi untuk Peter Norvig dan N-gram yaitu, 73% dan 79% sedangkan LSTM operasi *deletion* mencapai 90,8%.

## 5.2 Saran

Dari hasil penelitian yang telah dilakukan, dapat diambil beberapa saran sebagai berikut:

a. Dataset atau korpus yang digunakan bisa memasukan lebih banyak jenis kesalahaan ejaan.

b. *Spell correction* dapat mendeteksi kesalahan ejaan dalam sebuah kalimat panjang bukan hanya mendeteksi kata saja.
c. Mengoptimalkan nilai akurasi dan mempercepat waktu pemrosesan pada setiap metode *spell correction*.

# DAFTAR PUSTAKA


Ahmadzade, A., & Malekzadeh, S. (2021). *Spell Correction for Azerbaijani Language using Deep Neural Networks*.

Aouragh, S. L., Gueddah, H., Yousfi, A., Sidi, U., Abdellah, M. Ben, & Hicham, G. (2015). Adaptating the Levenshtein Distance to Contextual Spelling Correction ADAPTATING THE LEVENSHTEIN DISTANCE TO CONTEXTUAL SPELLING CORRECTION AOURAGH SI LHOUSSAIN. In *International Journal of Computer Science and Applications, ©Technomathematics Research Foundation* (Vol. 12, Issue 1). https://www.researchgate.net/publication/273758433

Aytan, B., & Sakar, C. O. (2023). Deep learning-based Turkish spelling error detection with a multi-class false positive reduction model. *Turk. J. Elec. Eng. & Comp. Sci.*, *31*(3), 581–595. https://doi.org/10.55730/1300-0632.4003

Chaabi, Y., & Ataa, F. A. (2022). Amazigh spell checker using Damerau-Levenshtein algorithm and N-gram. *Journal of King Saud University - Computer and Information Sciences*, *34*(8), 6116–6124. https://doi.org/10.1016/j.jksuci.2021.07.015

Christanti, M. V., Rudy, & Naga, D. S. (2018). Fast and accurate spelling correction using trie and Damerau-levenshtein distance bigram. *Telkomnika (Telecommunication Computing Electronics and Control)*, *16*(2), 827–833. https://doi.org/10.12928/TELKOMNIKA.v16i2.6890

Etoori, P., Chinnakotla, M., & Mamidi, R. (2018). *Automatic Spelling Correction for Resource-Scarce Languages using Deep Learning*. 146–152. https://github.com/PravallikaRao/SpellChecker

Fahma, A. I., Cholissodin, I., & Perdana, R. S. (2018). Identifikasi Kesalahan Penulisan Kata (Typographical Error) pada Dokumen Berbahasa Indonesia Menggunakan Metode N-gram dan Levenshtein Distance. *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer*, *2*(1), 53–62. https://doi.org/http://j-ptiik.ub.ac.id

Fahrudin, T. M., Sa'diyah, I., Latipah, Zahy, I., Illah, A., Lirna, C. C. B., & Acarya, B. S. (2021). *Seminar Nasional Informatika Bela Negara (SANTIKA) Aplikasi Pendeteksi Kesalahan Ejaan Bahasa Indonesia pada Karya Ilmiah Bidang Ilmu Komputer Menggunakan KEBI 1.0 Checker*. *2*.

Fahrudin, T. M., Sa'diyah, I., Latipah, Zahy, I., Illah, A., Lirna, C. C. B., & Acarya, S. B. (2022). A Rule-based Spelling Checker for Correcting Punctuation Errors in Indonesia

Text using KEBI 1.0 Checker. *International Seminar of Research Month 2021*, 1–8. https://doi.org/10.11594/nstp.2022.2433

Gipayana, M. (2017). Spell checker implementation to analyze the narrative essay of sixth-grade elementary school students in Indonesia. *Bulletin of Social Informatics Theory and Application*, *1*(1), 18–25. https://doi.org/10.31763/businta.v1i1.21

Hamidah, N., Yusliani, N., & Rodiah, D. (2020). Spelling Checker using Algorithm Damerau Levenshtein Distance and Cosine Similarity. In *Sriwijaya Journal of Informatic and Applications* (Vol. 01, Issue 01). http://sjia.ejournal.unsri.ac.id

Hardiyanti, M. (2021). Identifying The Common Type of Spelling Error by Leveraging Levenshtein Distance and N-gram. *Scientific Journal of Informatics*, *8*(1). https://doi.org/10.15294/sji.v8i1.xxxxx

Indah Ratnasari, C., Kusumadewi, S., & Rosita, L. (2017). A Non-Word Error Spell Checker for Patient Complaints in Bahasa Indonesia. In *International Journal of Information Technology* (Vol. 1, Issue 1). https://www.researchgate.net/publication/318927640

Jatminto, J., & Nuryana, I. K. D. (2016). Implementasi Spelling Checker dengan Algoritma Levenshtein distance pada Ensiklopedia IT (Information Technology) berbasis website. *Jurnal Ilmiah Inovasi Teknologi Informasi*, *1*(1).

Li, J., Xiao, W., & Zhang, C. (2023). Data security crisis in universities: identification of key factors affecting data breach incidents. *Humanities and Social Sciences Communications*, *10*(1). https://doi.org/10.1057/s41599-023-01757-0

Martin, R., Naga, D. S., & Mawardi, V. C. (2021). Penggunaan Spelling Correction Dengan Metode Peter Norvig dan N-gram. *Jurnal Ilmu Komputer Dan Sistem Informasi*, *9*(1), 175–180. https://doi.org/https://doi.org/10.24912/jiksi.v9i1.11591

Mutammimah, Sujaini, H., & Nyoto, R. D. (2017). Analisis Perbandingan Metode Spelling Corrector Peter Norvig dan Spelling Checker BK-Trees pada Kata Berbahasa Indonesia. *Jurnal Sistem Dan Teknologi Informasi*, *5*(1), 12–16.

Neto, A. F. de S., Bezerra, B. L. D., & Toselli, A. H. (2020). Towards the natural language processing as spelling correction for offline handwritten text recognition systems. *Applied Sciences (Switzerland)*, *10*(21), 1–29. https://doi.org/10.3390/app10217711

Purbaya, M. E., Rakhmadani, D. P., Maliana Puspa Arum, & Luthfi Zian Nasifah. (2023). Implementation of n-gram Methodology to Analyze Sentiment Reviews for Indonesian Chips Purchases in Shopee E-Marketplace. *Rekayasa Sistem Dan Teknologi Informasi*, *7*(3), 609–617. https://doi.org/10.29207/resti.v7i3.4726

Ramadhan, M. Z., & Lhaksmana, K. M. (2020). Improving Document Retrieval with Spelling Correction for Weak and Fabricated Indonesian-Translated Hadith. *Accredited by National Journal Accreditation*, *4*(2), 551–557. http://jurnal.iaii.or.id

Santoso, P., Yuliawati, P., Shalahuddin, R., & Wibawa, A. P. (2019). Damerau Levenshtein Distance for Indonesian Spelling Correction. *Jurnal Informatika*, *13*(2), 11. https://doi.org/10.26555/jifo.v13i2.a15698

Selvaraj, P. A., Jagadeesan, M., Harikrishnan, M., Vijayapriya, R., & Jayasudha, K. (2022). Survey on Spell Checker for Tamil Language Using Natural Language Processing. *Journal of Pharmaceutical Negative Results* ¦, *13*. https://doi.org/10.47750/pnr.2022.13.S07.026

Simanjuntak, M. S., Sujaini, H., & Safriadi, N. (2018). Spelling Corrector Bahasa Indonesia dengan Kombinasi Metode Peter Norvig dan N-Gram. *Jurnal Edukasi Dan Penelitian Informatika*, *4*(1), 17. https://doi.org/10.26418/jp.v4i1.24075

Soisoonthorn, T., Unger, H., & Maliyaem, M. (2023). Spelling Check: A New Cognition-Inspired Sequence Learning Memory. *Journal of Advances in Information Technology*, *14*(3), 399–410. https://doi.org/10.12720/jait.14.3.399-410

Walelign, S., Tesfaye, D., & Kebebew, T. (2021). Modelling and Designing of NLP Interface to Database for Afaan Oromoo. *International Journal of Innovative Research in Computer Science & Technology*, *9*(2), 6–13. https://doi.org/10.21276/ijircst.2021.9.2.2

Yulianto, M. M., Arifudin, R., & Alamsyah. (2018). Autocomplete and Spell Checking Levenshtein Distance Algorithm to Getting Text Suggest Error Data Searching in Library. *Scientific Journal of Informatics*, *5*(1), 2407–7658. http://journal.unnes.ac.id/nju/index.php/sji

# LAMPIRAN

Lampiran A : Daftar Kata Benar dan Salah

| No | Kata yang benar | Kata yang salah |
|---|---|---|
| 1 | Antrean | Antrian |
| 2 | Apotek | Apotik |
| 3 | Efisien | Efision |
| 4 | Resmi | Resmo |
| 5 | Fotokopi | Fotocopy |
| 6 | Diskon | Discon |
| 7 | Psikologi | Sikologi |
| 8 | Farmasi | Pharmasi |
| 9 | Higienis | Hijienis |
| 10 | Jurnalistik | Jurnalistis |
| 11 | Karir | Karier |
| 12 | Kreativitas | Kretivitas |
| 13 | Laboratorium | Laboratorim |
| 14 | Lisensi | Lesensi |
| 15 | Literatur | Literatir |
| 16 | Manajemen | Managemen |
| 17 | Metode | Metoda |
| 18 | Objektif | Obyektif |
| 19 | Otoritas | Autoritas |
| 20 | Pendaftaran | Pendaftar |
| 21 | Praktik | Praktek |
| 22 | Profesional | Profesionil |
| 23 | Resepsionis | Resipsionis |
| 24 | Sistematis | Sistemetis |
| 25 | Steril | Sterel |
| 26 | Strategi | Startegi |
| 27 | Subjektif | Subyektif |
| 28 | Teknis | Teknik |
| 29 | Terapi | Terapie |
| 30 | Variasi | Fariasi |

https://drive.google.com/drive/folders/1meL81zPo3iVxfWL0LNv-NLS8zIjh0IMU?usp=sharing