# Normalisasi Kata Tidak Baku yang Tidak Disingkat dengan Jarak Perubahan

I Gusti Bagus Baskara Nugraha[1], Rafi Dwi Rizqullah[1]

***Abstract*—Voice assistant technology is growing rapidly and its use has begun to spread to daily use. However, voice assistant usages are still limited to standard conversation languages. Meanwhile, Indonesian people are accustomed to informal language in daily conversation. This research gives solution to overcome the problem of voice assistants with informal words or words that will not be found in formal word dictionary. We propose text normalization using Levenshtein distance. Test result shows that normalization using Levenshtein distance outperform the normalization using Longest Common Subsequence (LCS) distance with accuracy difference of 8.34%.**

***Intisari*—Teknologi *voice assistant* (asisten suara) mulai berkembang pesat saat ini. Penggunaannya sudah mulai merambah kepada penggunaan sehari-hari. Namun, *voice assistant* masih terbatas pada penggunaan bahasa percakapan yang baku. Sementara itu, masyarakat Indonesia terbiasa mengucapkan bahasa tidak baku dalam percakapan sehari-hari. Makalah ini mencakup solusi untuk mengatasi permasalahan *voice assistant* dengan kata yang tidak baku atau tidak termasuk dalam kamus kata baku. Pendekatan yang digunakan sebagai solusi adalah melakukan normalisasi teks menggunakan jarak Levenshtein. Hasil pengujian menunjukkan bahwa normalisasi dengan jarak Levenshtein mengungguli normalisasi dengan jarak *Longest Common Subsequence* (LCS) dengan selisih akurasi sebesar 8,34%.**

***Kata Kunci*—*voice assistant*, kamus, kata tidak baku, normalisasi, jarak Levenshtein, jarak Jaro-Winkler.**

## I. PENDAHULUAN

*Voice assistant* (asisten suara) mulai menjadi tren yang makin berkembang. Gartner menyebutkan, pada tahun 2018, sebesar 30% interaksi manusia dengan teknologi melibatkan media yang mendukung percakapan dengan mesin pintar [1]. Berbicara dengan aplikasi *voice assistant* dapat menciptakan pengalaman yang lebih baik untuk pengguna. Sebanyak 41% pengguna *voice assistant* merasa bahwa berbicara dengan *voice assistant* terasa seperti berbicara dengan teman atau orang lain [2]. Pengalaman tersebut menjadikan pengguna merasa nyaman menggunakan *voice assistant* dalam kegiatan sehari-hari. Data dari Google menunjukkan 72% pengguna menggunakan aplikasi *voice assistant* sebagai bagian dari rutinitas harian mereka [2].

*Voice assistant* memiliki potensi untuk mempermudah kegiatan bagi masyarakat Indonesia. Namun, untuk dapat diterima oleh masyarakat, *voice assistant* harus mengatasi sebuah keadaan, yaitu hanya sekitar 20% masyarakat Indonesia yang berbicara dengan menggunakan bahasa Indonesia yang baku untuk percakapan sehari-hari [3]. Masyarakat Indonesia terbiasa menggunakan bahasa selain bahasa Indonesia untuk berbicara dengan kerabat dalam lingkungan tidak resmi, yakni dengan menggunakan campur kode. Campur kode adalah pemakaian dua bahasa atau lebih atau dua varian dari sebuah bahasa dalam satu masyarakat tutur [4]. *Voice assistant* harus dapat mengerti bahasa campur kode tersebut.

Namun, untuk dapat mengakomodasi campur kode tersebut, terdapat masalah pada model bahasa untuk *voice assistant*. Masalah tersebut adalah terdapat kata-kata campur kode yang tidak termasuk ke dalam kamus bahasa Indonesia yang baku. Sebagai contoh, kata "puter" yang biasa diartikan sebagai "putar". Padahal, model bahasa yang sudah tersedia menggunakan media-media resmi sebagai acuan untuk membentuk model bahasanya, seperti fastText milik Facebook dibangun dan dilatih dengan acuan kepada kata-kata dalam Wikipedia, termasuk untuk bahasa Indonesia [5]. Model bahasa fastText yang telah dilatih terdiri atas kamus dan nilai vektor untuk tiap kata di dalam kamus tersebut, sehingga penambahan beberapa kata ke dalam model bahasa tidak dimungkinkan.

Untuk mengatasi masalah ketiadaan kata dalam kamus, diperlukan sebuah metode berupa normalisasi kata, yaitu mengubah kata yang tidak baku menjadi kata baku dalam suatu bahasa. Terdapat sebuah penelitian mengenai pengembangan metode normalisasi teks pada suntingan Twitter bahasa Indonesia [6]. Namun, metode tersebut hanya digunakan pada lingkungan Twitter yang sebagian besar kata yang disunting merupakan kata tidak baku yang disingkat, tetapi tidak untuk kata tidak baku yang utuh atau tidak disingkat. Padahal, dalam *voice assistant* sebuah kata diucapkan tanpa ada penyingkatan kata.

Oleh karena itu, diperlukan sebuah metode normalisasi teks yang dapat digunakan dalam lingkungan kata yang tidak disingkat. Dengan begitu, metode normalisasi tersebut dapat digunakan untuk sistem *voice assistant* untuk bahasa Indonesia.

## II. JARAK DAN NORMALISASI

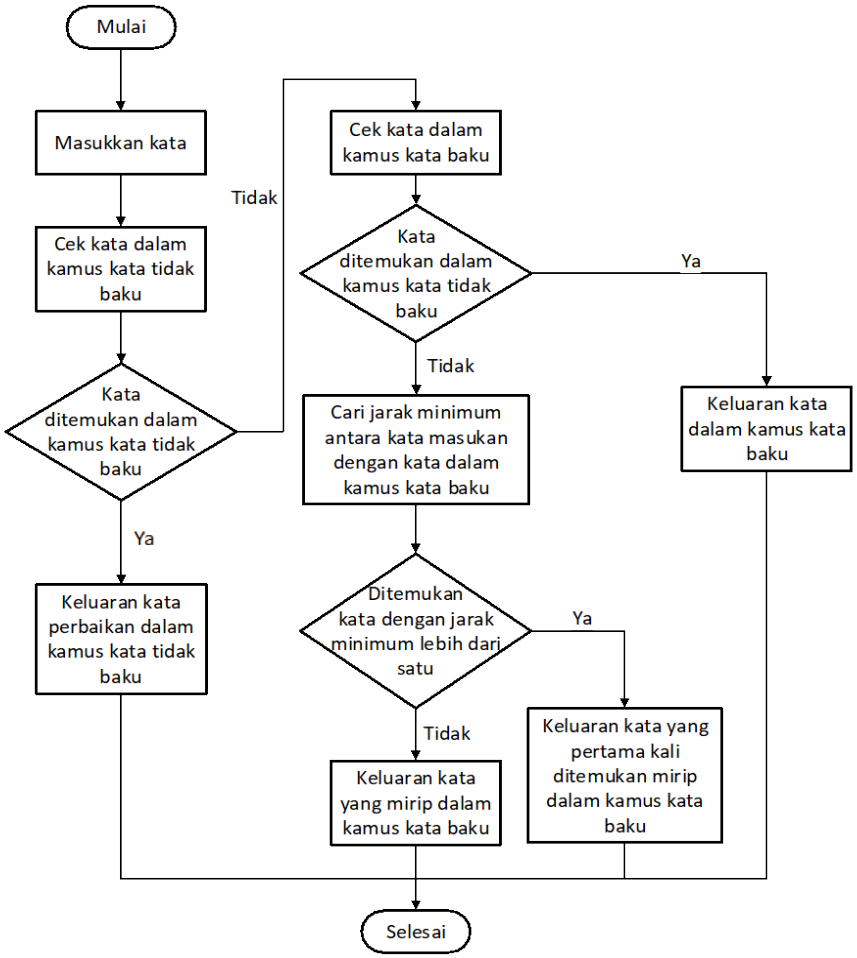
### A. Jarak Perubahan

Jarak perubahan adalah jumlah operasi perubahan terkecil yang dibutuhkan untuk mengubah teks, misalnya teks A menjadi teks B [7]. Terdapat beberapa variasi metode untuk mencari jarak perubahan kedua teks. Dengan kemampuan dalam menebak kata yang mirip, jarak perubahan dapat digunakan untuk melakukan normalisasi teks.

### B. Normalisasi Teks

Terdapat sebuah penelitian terkait normalisasi teks, yaitu penelitian normalisasi teks dalam suntingan media sosial

[1]*Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung, Jalan Ganesa 10, Bandung, INDONESIA (tlp: 022-2500985; e-mail: baskara@stei.itb.ac.id, rafidwiriz@gmail.com)*




Gbr. 1 Alur proses normalisasi teks [6].

Twitter, atau disebut juga dengan *tweet*, untuk bahasa Indonesia, yang memiliki karakteristik berupa kata yang disingkat [6]. Penelitian ini bertujuan untuk menerapkan algoritme penghitungan jarak dua kata dan membuat fungsi untuk normalisasi kata yang tidak baku menjadi kata baku.

*1) Alur Proses Normalisasi Teks:* Berdasarkan [6], alur proses normalisasi teks yang dibangun disajikan pada Gbr. 1. Alur tersebut merupakan alur untuk tiap kata yang terdapat dalam *tweet* berbahasa Indonesia. Proses membutuhkan dua jenis kamus, yaitu kamus kata tidak baku beserta perbaikannya dan kamus kata baku.

*2) Evaluasi Normalisasi Teks:* Evaluasi proses normalisasi teks dilakukan dengan memasukkan 200 kata tidak baku yang diambil dari kumpulan *tweet* berbahasa Indonesia yang telah diolah sebelumnya. Sebanyak sepuluh algoritme pengukuran jarak dievaluasi dengan mengukur persentase akurasi terhadap 200 kata *tweet* tidak baku. Penghitungan persentase akurasi dilakukan sebagai berikut:

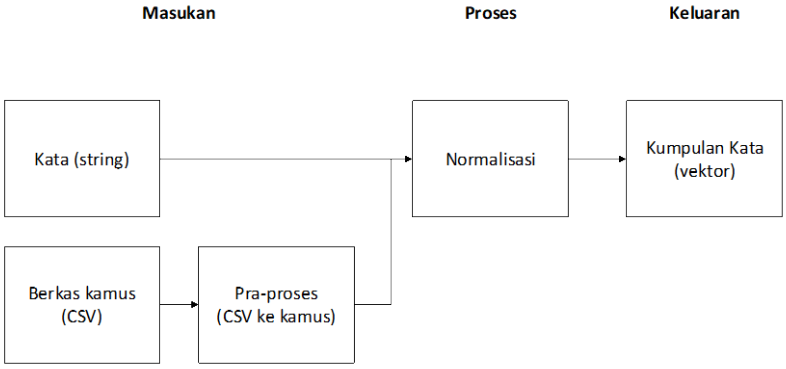
$$\text{Persentase akurasi} = \frac{x}{y} \times 100\%$$

dengan $x$ adalah jumlah kata yang tepat, yaitu kata tersebut sesuai dengan kata keluaran yang diharapkan, dan $y$ adalah jumlah seluruh kata yang diuji, yaitu 200 kata.

Hasil evaluasi menunjukkan bahwa algoritme jarak Jaro-Winkler unggul dalam pengujian pertama. Algoritme jarak Jaro-Winkler unggul dengan persentase akurasi sebesar 55,5%, diikuti dengan algoritme jarak *Longest Common Subsequence* (LCS) dengan akurasi sebesar 50%. Setelah dilakukan perbaikan terhadap kamus kata baku yang digunakan dalam penelitian, algoritme yang unggul berubah menjadi jarak LCS dengan persentase akurasi sebesar 69% diikuti oleh algoritme jarak Jaro-Winkler dengan akurasi sebesar 66,5%.

*3) Kelemahan:* Hasil evaluasi menunjukkan bahwa jarak LCS menempati urutan pertama dari sepuluh algoritme pengukuran jarak dengan persentase akurasi tertinggi. Meskipun demikian, jarak LCS hanya memiliki dua operasi perubahan saja, yaitu penambahan dan pengurangan karakter [7]. Tidak ada operasi untuk penggantian karakter dalam jarak LCS. Dengan begitu, untuk melakukan penggantian karakter dibutuhkan dua operasi yang digunakan secara bersamaan, yaitu operasi penambahan dan pengurangan karakter. Hal tersebut menyebabkan operasi penggantian karakter dalam jarak LCS memiliki nilai jarak perubahan sebesar dua dan dapat mengakibatkan tergusurnya kata perbaikan yang hanya memerlukan penggantian karakter dengan kata yang bertambah atau berkurang satu karakter saja.

Jarak LCS mendominasi persentase akurasi untuk pengujian terakhir dalam [6] sehingga jarak LCS cocok digunakan untuk




Gbr. 2 Alur metode normalisasi usulan.

TABEL I
CONTOH KELUARAN NORMALISASI TEKS DENGAN JARAK LEVENSHTEIN

| Contoh kata masukan | Contoh keluaran kumpulan kata | Contoh keluaran jarak terkecil |
|---|---|---|
| *mikir* | [‘pikir’, ‘bikir’, ‘kikir’, ‘likir’, ‘milir’, ‘zikir’] | 1 |
| *maen* | [‘main’, ‘man’, ‘men’] | 1 |

normalisasi teks dalam *tweet* berbahasa Indonesia, tetapi tidak cocok digunakan untuk kasus *voice assistant* yang sebagian besar kata tidak baku di dalamnya merupakan kata baku dengan penggantian karakter. Untuk menutupi kelemahan tersebut, diperlukan metode penghitungan jarak yang dapat menangani operasi penggantian karakter, beberapa di antaranya yaitu dengan menggunakan jarak Levenshtein dan jarak Jaro-Winkler.

### C. Jarak Levenshtein

Jarak Levenshtein, diciptakan oleh Vladimir Levenshtein, adalah jarak yang tercipta antara kedua teks atau kumpulan karakter yang dibandingkan. Jarak dihitung berdasarkan perubahan sebuah kumpulan karakter dengan kumpulan karakter yang lain, seperti penambahan, pengurangan, atau penggantian karakter [8]. Secara matematis, jarak Levenshtein didefinisikan sebagai (1).

$$\mathrm{lev}_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{jika } \min(i,j) = 0 \\ \min\begin{cases} \mathrm{lev}_{a,b}(i-1,j)+1 \\ \mathrm{lev}_{a,b}(i,j-1)+1 \\ \mathrm{lev}_{a,b}(i-1,j-1)+1_{(a\neq b)} \end{cases} & \text{jika } \min(i,j) \neq 0 \end{cases} \quad (1)$$

dengan $1_{(a\neq b)}$ adalah sebuah fungsi indikator yang bernilai 1 jika $a \neq b$ dan sebaliknya, dan $\mathrm{lev}_{a,b}(i,j)$ adalah jarak Levenshtein untuk karakter $i$ pertama pada teks $a$ dengan $j$ karakter pertama pada teks $b$.

### D. Jarak Jaro-Winkler

Jarak Jaro-Winkler merupakan jarak Jaro dengan tambahan oleh William E. Winkler [9]. Algoritme jarak Jaro menentukan kemiripan kedua teks dengan menghitung banyaknya karakter yang sesuai antara kedua teks dan posisi yang tidak terlalu jauh serta mengurangi banyak karakter yang sesuai tersebut dengan setengah dari banyaknya karakter yang mengalami transposisi [10]. Nilai kemiripan kedua teks berdasarkan Jaro didefinisikan sebagai (2) [11].

$$sim_j = \begin{cases} 0 & \text{jika } m = 0 \\ \frac{1}{3}\left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m}\right) & \text{jika } m \neq 0 \end{cases} \quad (2)$$

dengan $sim_j$ adalah nilai kemiripan Jaro dalam nilai desimal dengan rentang nilai 0 hingga 1, $|s_i|$ adalah panjang dari teks $s_i$ , $m$ adalah jumlah karakter yang sesuai, serta $t$ adalah setengah dari jumlah karakter sesuai yang mengalami transposisi. Karakter yang sama dalam $s_1$ dan $s_2$ dianggap sesuai jika selisih posisinya tidak lebih dari $\left\lfloor \frac{\max(|s_1|,|s_2|)}{2} \right\rfloor - 1$.

Winkler menambahkan algoritme tersebut dengan menerapkan penalti terhadap karakter yang tidak sesuai dalam empat karakter pertama [10]. Nilai kemiripan Jaro-Winkler didefinisikan sebagai (3) [9].

$$sim_w = sim_j + \ell p\left(1 + sim_j\right) \quad (3)$$

dengan $sim_w$ adalah nilai kemiripan Jaro-Winkler dalam nilai desimal dengan rentang nilai 0 hingga 1, $sim_j$ adalah nilai kemiripan Jaro dari (2), $\ell$ adalah panjang dari *common string* dengan nilai maksimal yaitu empat, dan $p$ adalah faktor skala yang ditentukan oleh pengguna.

## III. NORMALISASI TEKS DENGAN JARAK PERUBAHAN

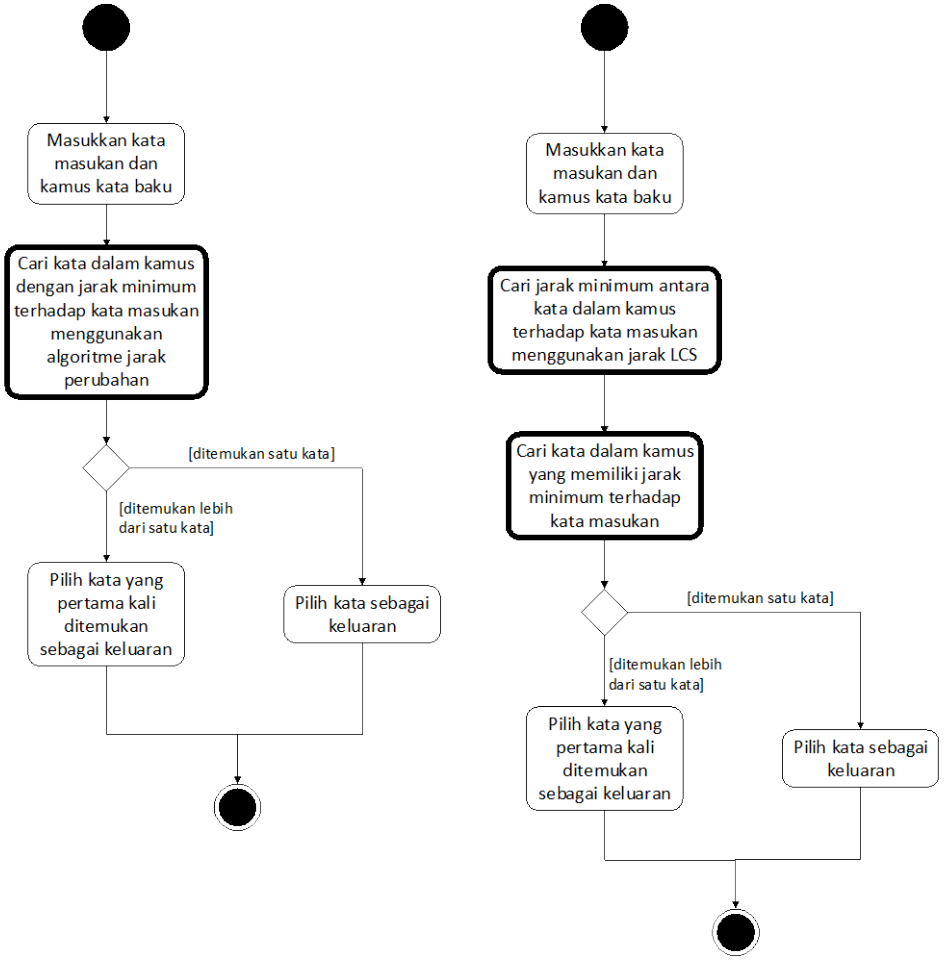
Rancangan untuk keseluruhan metode normalisasi teks usulan terdapat dalam Gbr. 2. Metode normalisasi teks usulan memiliki tiga bagian utama, yaitu bagian masukan, proses, dan keluaran.

### A. Masukan Normalisasi Teks

Metode normalisasi usulan memiliki dua jenis masukan, yaitu kata dan kamus kata baku. Kedua masukan diperlukan sehingga metode normalisasi dapat mencari kemiripan antara kata masukan dengan kamus kata baku.

### B. Proses Normalisasi Teks

Gbr. 3(a) menunjukkan alur proses untuk rancangan metode normalisasi yang diusulkan. Metode normalisasi usulan menggunakan jarak Levenshtein sebagai penghitungan jarak antara kedua teks.




(a) (b)

Gbr. 3 Alur proses normalisasi teks, (a) dengan metode usulan, (b) dengan metode [6].

Terdapat perbedaan alur proses antara metode normalisasi [6] dengan metode normalisasi usulan. Perbedaan tersebut dapat dilihat pada blok proses dengan garis tepi tebal pada Gbr. 3. Perbedaan pertama terletak pada penggantian algoritme jarak LCS dengan algoritme usulan, yaitu jarak Levenshtein. Selain itu, metode usulan dibandingkan juga dengan jarak Jaro-Winkler yang populer. Kedua, metode normalisasi usulan mempersingkat proses normalisasi dengan menemukan kumpulan kata prediksi dengan jarak minimum secara langsung dibandingkan metode normalisasi [6] yang harus mencari jarak minimum terlebih dahulu kemudian menemukan kumpulan kata prediksi dengan jarak tersebut.

### C. *Keluaran Normalisasi Teks*

Contoh keluaran metode normalisasi teks usulan tertera pada Tabel I. Salah satu contoh yang digunakan adalah kata "*mikir*". Kata "*mikir*", berdasarkan jarak Levenshtein, memiliki keluaran kumpulan kata-kata baku, yaitu "pikir", "bikir", "kikir", "likir", "milir", dan "zikir". Semua kata baku tersebut memiliki jarak perubahan yang sama dengan kata "*mikir*" yaitu 1. Jarak tersebut juga merupakan jarak terkecil yang dapat dicapai antara kata masukan dengan seluruh kata yang ada di dalam kamus.

## IV. PENGUJIAN

Bagian ini membahas pengujian normalisasi teks pada metode normalisasi teks [6], metode normalisasi teks denganjarak Jaro-Winkler, dan metode normalisasi teks dengan jarak Levenshtein. Pembahasan yang tertera dalam pengujian ini meliputi perangkat dan lingkungan yang digunakan untuk pengujian, metode pengujian, skenario pengujian, serta hasil dan analisis pengujian.

### A. *Metode Pengujian*

Pengujian dilakukan dengan melakukan perbandingan antara metode normalisasi teks [6], metode normalisasi dengan jarak Jaro-Winkler, dan metode normalisasi teks dengan jarak Levenshtein. Ketiga metode normalisasi diberi masukan yang sama. Keluaran dari ketiga metode dinilai dengan menggunakan persentase akurasi yang disesuaikan dengan skenario yang berbeda. Penilaian persentase akurasi dilakukan dengan menggunakan:

$$\text{Persentase akurasi} = \frac{x}{y} \times 100\%$$

TABEL II
LIMA KATA PERTAMA DALAM DATA UJI BESERTA PERBAIKANNYA

| Kata tidak baku | Perbaikan kata |
|---|---|
| *abadiat* | abadiah |
| *abdul* | abdu |
| *abimana* | abaimana |
| *ablur* | hablur |
| *adan* | azan |

TABEL III
LIMA BELAS KATA PERTAMA DALAM KAMUS KATA BAKU

| Kamus Kompas-KBBI | | |
|---|---|---|
| yang | dengan | pada |
| di | untuk | tidak |
| dan | dari | juga |
| ini | dalam | ke |
| itu | akan | tersebut |

dengan $x$ adalah jumlah kata yang sesuai dengan perbaikan kata dan $y$ adalah jumlah seluruh kata yang termasuk dalam data uji [6].

*1) Masukan:* Terdapat dua masukan yang termasuk dalam pengujian, yaitu data uji berisi 2.398 kata serta kamus kata baku berisi 29.194 kata.

Data uji yang digunakan dalam pengujian diambil dari kumpulan kata tidak baku yang berada dalam KBBI [12]. Tabel II menunjukkan lima contoh data yang digunakan dalam pengujian. Data uji terdiri atas kata tidak baku dan perbaikan dari kata tidak baku tersebut menjadi kata baku. Sebagai catatan, semua kata dalam kata uji merupakan kata-kata yang tidak disingkat.

Kamus kata baku yang digunakan berasal dari penggabungan antara kamus dalam perangkat lunak KBBI *offline* [12] dan kamus dari korpus arsip Kompas tahun 2012 [13]. Kedua kamus tersebut dibersihkan dari kata-kata tidak baku, lalu seluruh kata dalam kamus diurutkan berdasarkan kamus korpus Kompas. Tabel III menunjukkan lima belas kata pertama yang ada dalam kamus kata baku.

*2) Keluaran:* Keluaran dari ketiga metode normalisasi adalah sebuah vektor yang berisi kumpulan kata prediksi dalam kamus kata baku yang memiliki jarak paling kecil dari kata dalam data uji. Tabel IV menunjukkan contoh keluaran yang muncul jika ketiga metode normalisasi diberi masukan tertentu.

### *B. Skenario Pengujian*

Penilaian akurasi keluaran ketiga metode normalisasi ditentukan oleh skenario pengujian yang dilakukan. Terdapat dua skenario yang dijalankan dalam pengujian ini. Pada skenario pertama, penilaian dilakukan dengan melihat kesesuaian antara perbaikan kata dengan kata pertama dalam kumpulan kata prediksi, sedangkan dalam skenario kedua penilaian dilakukan dengan melihat terdapat perbaikan kata di dalam kumpulan kata prediksi atau tidak.

Gbr. 4 menunjukkan perbedaan antara skenario pertama dengan skenario kedua. Jika skenario pertama hanya memeriksa urutan pertama dari kumpulan kata prediksi, skenario kedua memeriksa seluruh kumpulan kata prediksi hingga ditemukan kata yang sesuai dengan perbaikan kata.

TABEL IV
CONTOH KELUARAN KETIGA METODE TERHADAP MASUKAN KATA

| Masukan | Keluaran [6] | Keluaran Jaro-Winkler | Keluaran Levenshtein |
|---|---|---|---|
| abadiat | [‘abadi’, ‘abadiah’, ‘ahadiat’, ‘baiat’] | [‘abadi’, ‘abadiah’] | [‘abadiah’, ‘ahadiat’] |
| abdul | [‘abdu’, ‘abul’] | [‘abdu’] | [‘abdu’, ‘abul’] |
| abimana | [‘abaimana’] | [‘abian’] | [‘abaimana’] |

TABEL V
HASIL AKURASI UNTUK PENGUJIAN KETIGA METODE NORMALISASI DALAM SKENARIO PERTAMA

| Metode | Jumlah kata benar | Akurasi |
|---|---|---|
| Metode [6] | 622 | 25,93% |
| Metode Jaro-Winkler | 602 | 25,10% |
| Metode Levenshtein | 774 | 32,28% |

TABEL VI
HASIL AKURASI UNTUK PENGUJIAN KETIGA METODE NORMALISASI DALAM SKENARIO KEDUA

| Metode | Jumlah kata benar | Akurasi |
|---|---|---|
| Metode [6] | 1.233 | 51,42% |
| Metode Jaro-Winkler | 715 | 29,82% |
| Metode Levenshtein | 1.686 | 70,31% |

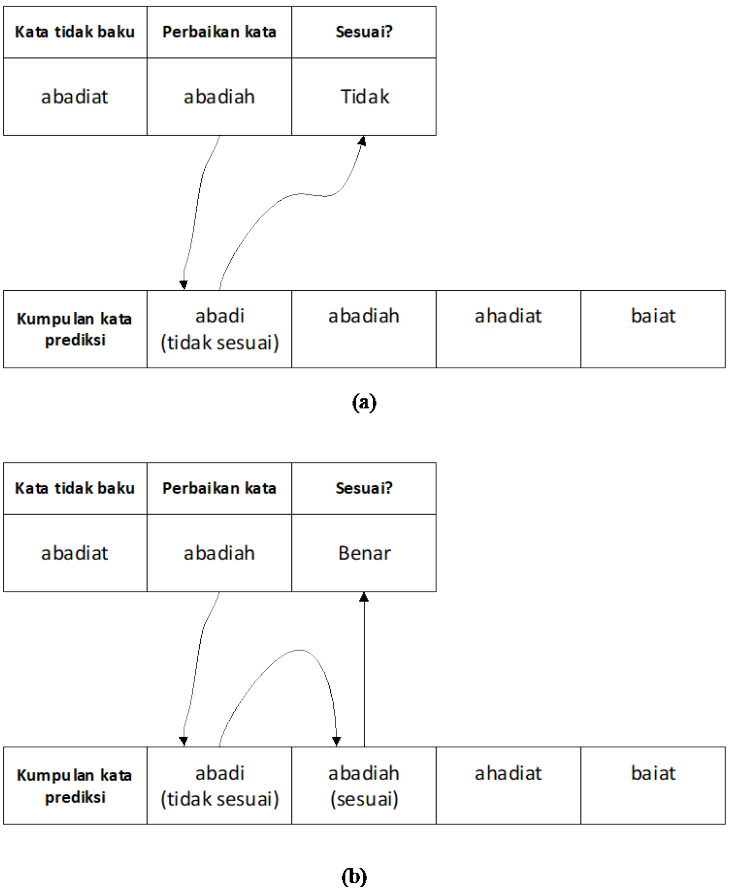
### *C. Hasil Pengujian*

*1) Hasil Pengujian Skenario Pertama:* Hasil pengujian untuk skenario pengujian ditunjukkan pada Tabel V. Hasil pengujian menunjukkan bahwa metode normalisasi yang diusulkan mengungguli kedua metode normalisasi yang lain dengan selisih akurasi dengan metode [6] sebesar 6,35%. Metode normalisasi dengan Jaro-Winkler menempati peringkat terakhir dengan selisih akurasi dengan metode normalisasi [6] sebesar 0,83%.

Hasil metode normalisasi Jaro-Winkler yang berada di posisi terakhir disebabkan oleh teknik Jaro-Winkler dalam menentukan kemiripan antara kedua kata. Jaro-Winkler menganggap bahwa kedua kata dikatakan mirip jika terdapat karakter yang sesuai dengan jumlah yang banyak serta tidak ada transposisi dengan karakter yang sesuai tersebut. Maka, Jaro-Winkler akan menganggap kata dengan penambahan satu karakter memiliki kemiripan yang lebih tinggi dibandingkan kata dengan penggantian satu karakter saja, mengakibatkan kata yang mengalami penggantian karakter tergusur dari pencarian kata baku.

Meskipun demikian, hasil akurasi pengujian untuk ketiga metode masih menunjukkan angka di bawah 50%. Hal tersebut disebabkan oleh perbaikan kata uji bukan merupakan kata yang berada dalam urutan pertama kumpulan kata prediksi. Hal tersebut berarti perbaikan kata uji bukan merupakan kata yang sering digunakan dalam korpus Kompas.

*2) Hasil Pengujian Skenario Kedua:* Hasil pengujian untuk skenario kedua disajikan pada Tabel VI dan hasil untuk kedua




Gbr. 4 Perbedaan antar skenario, (a) skenario pertama, (b) skenario kedua.

```
100%|██████████████████████████████
██| 2398/2398 [2:47:33<00:00,  3.34s/it]

Skenario1, LCS benar:  622
Skenario1, LV benar:  774
Skenario1, JW benar:  602

Skenario2, LCS benar:  1233
Skenario2, LV benar:  1686
Skenario2, JW benar:  715
```

Gbr. 5 Seluruh hasil pengujian ketiga metode dengan menggunakan Jupyter Notebook.

pengujian diperlihatkan pada Gbr. 5. Hasil pengujian tersebut menunjukkan peningkatan yang signifikan untuk selisih antara metode normalisasi Levenshtein dengan kedua metode normalisasi yang lain. Hasil akurasi untuk metode normalisasi Levenshtein masih mengungguli metode normalisasi [6] dengan selisih sebesar 18,89%. Metode normalisasi dengan Jaro-Winkler masih tetap berada di posisi terakhir dengan selisih akurasi 21,6% dari metode normalisasi [6].

Hasil akurasi metode normalisasi dengan Jaro-Winkler yang masih di bawah 50% masih disebabkan oleh satuan penilaian kemiripan yang digunakan oleh Jaro-Winkler. Satuan penilaian yang digunakan adalah bilangan desimal. Bilangan tersebut lebih sensitif terhadap perubahan dibandingkan dengan bilangan bulat. Hal tersebut mengakibatkan kumpulan kata dengan kemiripan tertinggi sementara tergusur dengan mudah oleh kata baru dengan kemiripan yang lebih tinggi.

Meskipun hasil akurasi bertambah dibandingkan sebelumnya, persentase akurasi belum menyentuh angka 100%. Hal tersebut dapat disebabkan oleh jarak perubahan antara kata tidak baku dan perbaikan kata uji tidak termasuk dalam kumpulan kata dengan jarak perubahan minimum. Beberapa kata perbaikan dalam data uji memiliki jarak yang lebih besar dari jarak minimum yang ditemukan antara kata dalam data uji dengan beberapa kata dalam kamus kata baku.

Hasil pengujian tersebut menunjukkan bahwa masalah yang terjadi jika perbaikan kata uji telah diselesaikan, jika algoritme dalam metode normalisasi dapat melihat seluruh kata yang terkandung pada kumpulan kata prediksi. Namun, algoritme tersebut hanya dapat melihat kumpulan kata. Algoritme tidak dapat menentukan kata yang tepat digunakan, jika algoritme hendak diterapkan dalam sistem *voice assistant*. Untuk penerapan dalam sistem *voice assistant*, metode normalisasi juga harus dapat menentukan kata dalam kumpulan kata prediksi yang tepat digunakan.

### *D. Kesimpulan Pengujian*

Pengujian yang telah dilakukan menghasilkan kesimpulan bahwa metode normalisasi dengan jarak Levenshtein lebih cocok digunakan sebagai normalisasi kata tidak baku yang tidak disingkat dibandingkan dengan metode normalisasi dengan jarak Jaro-Winkler. Hal tersebut ditunjukkan dengan

unggulnya metode normalisasi dengan Levenshtein dalam semua skenario pengujian, sedangkan metode normalisasi dengan Jaro-Winkler tidak mampu mengungguli metode normalisasi [6].

## V. KESIMPULAN

Telah dibuat sebuah metode normalisasi teks menggunakan algoritme jarak Levenshtein. Pengujian salah satu skenario menunjukkan bahwa metode normalisasi dengan menggunakan jarak Levenshtein unggul dalam akurasi dibandingkan dengan metode normalisasi sebelumnya yang menggunakan jarak LCS, dengan selisih persentase akurasi sebesar 8,34%. Setelah itu, dilakukan pengujian dengan skenario yang lain. Pengujian tersebut menunjukkan metode normalisasi dengan jarak Levenshtein masih tetap unggul dengan selisih persentase akurasi dengan metode normalisasi sebelumnya sebesar 18,89%. Hal tersebut menunjukkan bahwa metode normalisasi dengan jarak Levenshtein lebih cocok digunakan dalam sistem *voice assistant* karena algoritme jarak Levenshtein mendukung operasi penggantian karakter dibandingkan algoritme jarak LCS dan algoritme jarak Jaro-Winkler.

Saran-saran yang diberikan untuk penelitian selanjutnya adalah mencari solusi untuk kasus perbaikan kata yang tidak termasuk dalam kumpulan kata prediksi dengan jarak minimum serta menentukan algoritme yang dapat digunakan untuk menentukan kata dalam kumpulan kata prediksi yang tepat digunakan.

## REFERENSI


[1] M. Escherich dan W. Goertz. (2015) “Market Trends: Voice as a UI on Consumer Devices—What Do Users Want?” [Online], https://www.gartner.com/doc/3021226/market-trends-voice-ui-consumer/, tanggal akses: 2-Nov-2017.

[2] S. Kleinberg (2018) “5 ways voice assistance is shaping consumer behavior,” [Online], https://thinkwithgoogle.com/consumer-insights/voice-assistance-consumer-experience/, tanggal akses: 30-Jul-2018.

[3] A. Na’im dan H. Syaputra, *Kewarganegaraan, Suku Bangsa, Agama dan Bahasa Sehari-hari Penduduk Indonesia: Hasil Sensus Penduduk 2010*, Sumarwanto dan T. Irianto, Ed. Jakarta, Indonesia: Badan Pusat Statistik, 2012.

[4] A. Chaer dan L. Agustina, *Sosiolinguistik: Suatu Pengantar*. Jakarta-Indonesia: Rineka Cipta, 1995.

[5] P. Bojanowski, E. Grave, A. Joulin, dan T. Mikolov, “Enriching Word Vectors with Subword Information,” *Transactions of the Association for Computational Linguistics*, Vol. 5, hal. 135–146, 2017.

[6] T.S. Saragih, “Normalisasi Teks pada Teks Twitter Berbahasa Indonesia menggunakan Algoritme Jarak String pada R”, Skripsi, Institut Teknologi Bandung, Bandung, Indonesia, 2017.

[7] H. Schütze, C.D. Manning, dan P. Raghavan, *Introduction to Information Retrieval*. Cambridge, UK: Cambridge University Press, 2008.

[8] V.I. Levenshtein, “Binary Codes Capable of Correcting Deletions, Insertions, and Reversals,” *Soviet Physics Doklady*, Vol. 10, hal. 707-710, 1966.

[9] W.E. Winkler, “String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage,” *Proceedings of the Section on Survey Research*, 1990, hal. 354-359.

[10] M.P. Van der Loo, “The Stringdist Package for Approximate String Matching,” *The R Journal*, Vol. 6, hal. 111–122, 2014.

[11] M.A. Jaro, “Advances in Record-Linkage Methodology as Applied to Matching the 1985 Census of Tampa, Florida,” *Journal of the American Statistical Association*, Vol. 84, hal. 414–420, 1989.

[12] (2019) KBBI Offline Remake with Qt, [Online], https://github.com/bgli/kbbi-qt, tanggal akses: 1-Feb-2019.

[13] I. Lanin, J. Geovedi, dan W. Soegijoko, “Perbandingan Distribusi Frekuensi Kata Bahasa Indonesia di Kompas, Wikipedia, Twitter, dan Kaskus,” *KOLITA 11: Konferensi Linguistik Tahunan Atma Jaya Kesebelas*, 2013, hal. 249–252.