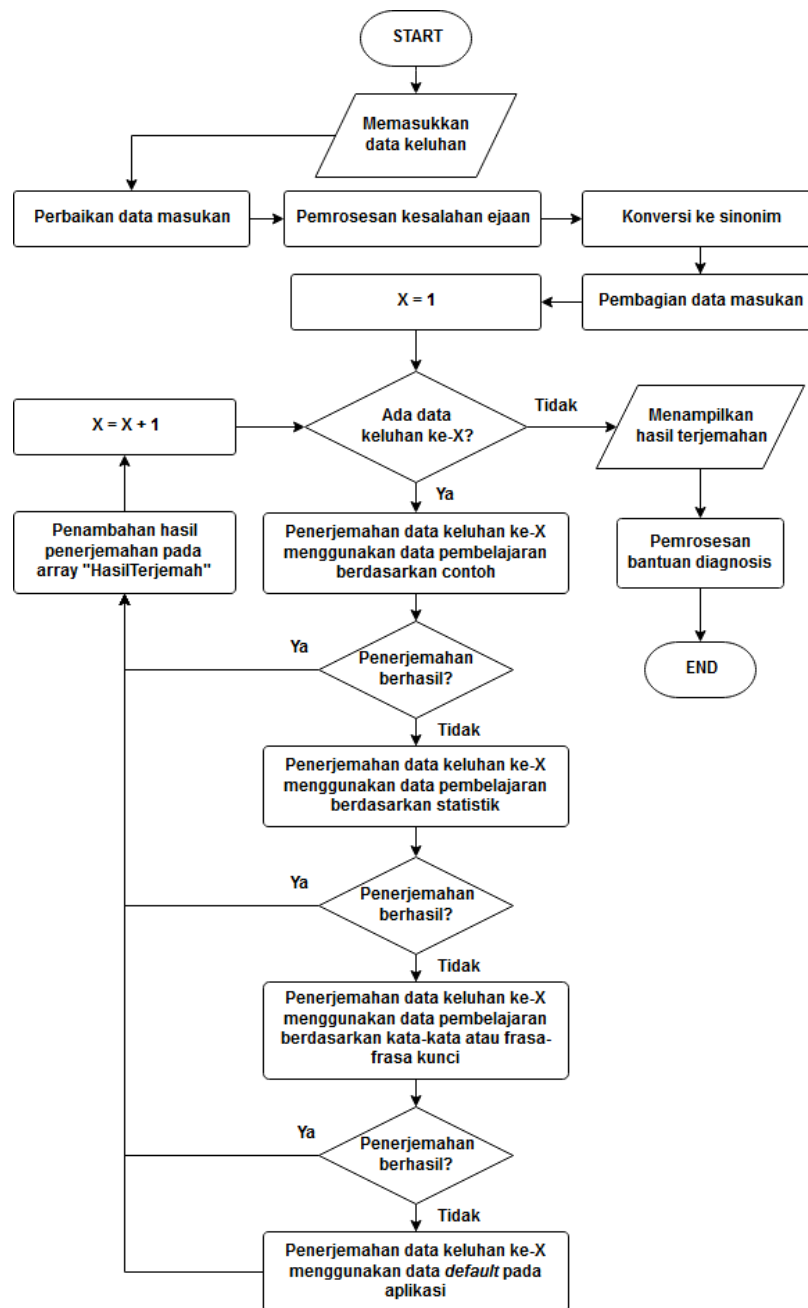


BAB IV

PEMODELAN SISTEM

4.1. Gambaran Model

Gambaran model keseluruhan proses penerjemahan data masukan keluhan pasien pada aplikasi ini dapat dilihat pada Gambar 4.1. di bawah ini:



Gambar 4.1. Model penerjemahan masukan keluhan pasien.

Berikut ini adalah penjelasan dari Gambar 4.1. di atas:

a. Masukan Data Keluhan Pasien

Aplikasi mampu menerima masukan dalam bentuk teks atau suara. Untuk masukan dalam bentuk teks dapat diketikkan secara langsung pada antarmuka aplikasi atau dengan mengimpor *file* teks (.txt) atau *file* excel (.xlsx) yang tersimpan pada media penyimpanan yang berisi data teks keluhan pasien dengan format yang sudah ditentukan. Untuk masukan dalam bentuk suara dapat dimasukkan langsung menggunakan mikrofon atau dengan mengimpor *file* suara dalam format WAV atau format MP3 yang nantinya dikonversi dahulu menggunakan aplikasi *open source* FFmpeg ke format WAV secara otomatis oleh aplikasi.

Fitur *speech recognition* dengan menggunakan Sphinx4 API digunakan untuk mengubah masukan dalam bentuk suara ke bentuk teks. Pada fitur *speech recognition* ini dilakukan analisis tingkat fonologi untuk mengenali fon-fon yang diucapkan oleh *user*. Kata-kata yang diucapkan oleh *user* dibagi menjadi beberapa *frame* kemudian tiap-tiap *frame* tersebut dicocokkan dengan model-model trifon yang ada pada model akustik. Model trifon yang paling sesuai dengan kata-kata yang diucapkan oleh *user* dipilih sebagai hasil transkripsi. Model bahasa pada fitur *speech recognition* kemudian melakukan analisis sintaksis untuk memilih rangkaian kata yang benar atau memiliki probabilitas paling tinggi pada proses transkripsi suara tersebut. Misalnya bigram “tubuh sakit” memiliki probabilitas lebih tinggi terdapat dalam teks Bahasa Indonesia ketimbang bigram “tujuh sakit”, sehingga jika *user* mengucapkan bigram “tubuh sakit” maka aplikasi kemungkinan besar tidak akan keliru mentranskripsikannya menjadi “tujuh sakit”, begitu pula trigram “buang air besar” memiliki probabilitas lebih tinggi terdapat dalam teks Bahasa Indonesia ketimbang trigram “buang akhir besar”, sehingga jika *user* mengucapkan trigram “buang air besar” maka aplikasi kemungkinan besar tidak akan keliru mentranskripsikannya menjadi “buang akhir besar”

b. Perbaikan Data Masukan

Data teks keluhan pasien yang sudah dimasukkan kemudian diperbaiki dahulu isinya untuk mempermudah pemrosesan di tahap berikutnya. Perbaikan pada data teks keluhan pasien antara lain pengubahan seluruh huruf pada teks ke bentuk huruf kecil (*case folding*), penghapusan kata-kata partikel yang tidak diperlukan dalam proses penerjemahan (misal kata loh, ah, wah, duh, dll.), perbaikan penulisan singkatan dan kata ulang agar

sesuai dengan tata bahasa baku Bahasa Indonesia, perbaikan penulisan tanda baca, penghapusan baris kosong, dan penghapusan spasi-spasi berlebihan termasuk spasi-spasi di awal dan di akhir kalimat (*trimming*). Pada tahap ini dilakukan analisis morfologi untuk mengubah penulisan singkatan dan kata ulang yang tidak baku menjadi lebih baku, misal penulisan kata ulang “paru2” diubah menjadi kata ulang “paru-paru” dan penulisan kata “jari jari” diubah menjadi kata “jari-jari”.

c. Pengoreksian Kesalahan Ejaan/Penulisan

Setelah isi teks masukan diperbaiki kemudian dilakukan pencarian kesalahan ejaan pada kata-kata yang terdapat di dalam teks masukan tersebut. Hanya kata-kata dengan jumlah karakter lima atau lebih yang akan dicari kesalahan ejaannya. Kesalahan ejaan yang dicari oleh aplikasi antara lain berupa kelebihan satu huruf seperti kata “makan” salah tulis menjadi “makann”, kekurangan satu huruf seperti kata “sakit” salah tulis menjadi “akit”, atau kesalahan menulis sebuah huruf menjadi huruf lainnya seperti kata “pusing” salah tulis menjadi “puting”. Pada proses pengoreksian kesalahan ejaan ini, setiap kata yang terdapat pada teks masukan dicek dahulu apakah ada pada daftar kata-kata yang dikenali oleh aplikasi, jika tidak ada maka aplikasi menganggap bahwa kata tersebut salah tulis dan akan menampilkan rekomendasi kata yang sesuai kepada *user*. Rekomendasi kata didapatkan dengan mencari kata-kata pada *database* aplikasi yang paling mirip dengan kata yang dianggap salah oleh aplikasi tersebut, yaitu mencari kata yang hanya berbeda satu huruf saja dengan kata yang dianggap salah oleh aplikasi tersebut. Jika tidak ditemukan kata yang mirip pada *database* maka aplikasi tidak akan menampilkan rekomendasi kata kepada *user*. Jika ada lebih dari satu kata yang mirip maka dipilih kata yang probabilitas kemunculannya dalam teks Bahasa Indonesia serta dalam teks keluhan pasien yang paling tinggi. Setelah aplikasi menampilkan rekomendasi kata yang sesuai kepada *user*, *user* dapat mengklik rekomendasi kata tersebut untuk mengubah seluruh kata yang dianggap salah tersebut menjadi rekomendasi kata oleh aplikasi.

d. Konversi ke Sinonim

Proses konversi kata ke sinonimnya dilakukan untuk mencegah dua kata yang memiliki makna yang sama dianggap sebagai kata yang berbeda oleh aplikasi. Misalnya keluhan “sakit telinga” memiliki makna yang sama dengan keluhan “sakit kuping”, sehingga semua kata “kuping” pada teks masukan diubah terlebih dahulu menjadi kata “telinga” oleh aplikasi sebelum dilakukan pemrosesan pada tahap selanjutnya. Proses

konversi kata ke sinonimnya ini juga dilakukan untuk mengubah kata-kata tidak baku menjadi kata-kata yang lebih baku, mengubah kata-kata dalam bahasa daerah menjadi kata-kata dalam Bahasa Indonesia, dan mengubah penulisan bilangan dalam bentuk teks seperti kata “seratus” menjadi penulisan bilangan dalam bentuk angka seperti “100”.

e. Pembagian Data Masukan

Sebelum dilakukan proses penerjemahan, data masukan tersebut dibagi-bagi terlebih dahulu menjadi beberapa data masukan untuk pasien berbeda dan dibagi kembali menjadi beberapa data keluhan untuk masing-masing pasien tersebut. Aplikasi mampu menerima data teks masukan untuk beberapa pasien berbeda secara sekaligus. Teks masukan untuk tiap-tiap pasien berbeda dipisahkan oleh baris baru. Keluhan-keluhan berbeda pada setiap pasien dipisahkan oleh tanda titik (.). Tanda koma (,), kata “dan”, dan kata “serta” terkadang juga dapat memisahkan keluhan yang berbeda, dan proses pengecekan apakah tanda koma (,), kata “dan”, dan kata “serta” ini memisahkan keluhan yang berbeda dilakukan pada tahap penerjemahan.

f. Penerjemahan Data Keluhan

Proses penerjemahan dilakukan dalam empat tahapan. Berdasarkan urutannya, tahap pertama adalah proses penerjemahan menggunakan data pembelajaran berdasarkan contoh, tahap kedua adalah proses penerjemahan menggunakan data pembelajaran berdasarkan statistik, tahap ketiga adalah proses penerjemahan menggunakan data pembelajaran berdasarkan kata-kata atau frasa-frasa kunci, dan tahap terakhir adalah proses penerjemahan menggunakan data *default* pada aplikasi. Jika *user* tidak melakukan proses pembelajaran pada aplikasi, maka aplikasi akan melakukan proses penerjemahan menggunakan data *default*. Pada proses ini aplikasi akan mencari kata-kata kunci, frasa-frasa kunci, dan lokasi kemunculannya untuk mencari hasil terjemahannya yang sesuai berdasarkan data pada *database*. Pada proses penerjemahan ini dilakukan analisis sintaksis untuk melakukan *parsing* terhadap masukan keluhan supaya diketahui urutan kata-kata yang benar dalam proses penerjemahan. Misalnya kalimat “Menstruasi tidak teratur.” menunjukkan suatu keluhan, namun kalimat “Teratur tidak menstruasi.” tidak menunjukkan suatu keluhan karena urutan kata-katanya tidak sesuai dengan aturan tata bahasa. Analisis sintaksis juga digunakan untuk menerjemahkan keluhan yang mengandung tanda koma (,), kata “dan”, atau kata “serta”. Misalnya pada kalimat “kaki sakit, dan kesemutan”, aplikasi memisahkan kalimat tersebut menjadi dua keluhan yaitu

“kaki sakit” dan “kaki kesemutan”, pada keluhan kedua diketahui bahwa organ yang mengalami kesemutan adalah kaki setelah melihat kata-kata sebelum tanda koma. Untuk pencarian waktu awal kemunculan keluhan, frekuensi keluhan, dan sifat keluhan, dilakukan proses pencarian menggunakan ekspresi reguler.

User dapat melakukan proses pembelajaran terhadap aplikasi agar tingkat akurasi penerjemahan oleh aplikasi semakin baik. Ada tiga metode pembelajaran yang dapat dilakukan terhadap aplikasi, yaitu pembelajaran untuk penerjemahan berdasarkan contoh, pembelajaran untuk penerjemahan berdasarkan kata-kata atau frasa-frasa kunci, dan pembelajaran untuk penerjemahan berdasarkan statistik. Pada pembelajaran untuk penerjemahan berdasarkan contoh, *user* memasukkan contoh masukan keluhan beserta hasil terjemahannya ke dalam *database* aplikasi, nantinya jika *user* memasukkan keluhan yang sama dengan salah satu contoh masukan yang ada pada *database* maka aplikasi akan menampilkan terjemahan yang sesuai berdasarkan contoh masukan tersebut. Misal jika *user* memberikan contoh masukan “dahi panas” dengan hasil terjemahan X untuk proses pembelajaran berdasarkan contoh, maka jika nanti *user* memasukkan *input* “dahi panas” atau “jidat panas”, aplikasi akan menampilkan hasil terjemahan X.

Pada pembelajaran untuk penerjemahan berdasarkan kata-kata atau frasa-frasa kunci, *user* memasukkan dua atau tiga kata-kata atau frasa-frasa kunci yang harus ada pada sebuah teks masukan keluhan beserta hasil terjemahan dari teks masukan keluhan tersebut ke dalam *database* aplikasi, nantinya jika *user* memasukkan keluhan yang berisi kata-kata atau frasa-frasa kunci yang sama dengan yang ada pada sebuah entri di dalam *database* aplikasi maka aplikasi akan menampilkan terjemahan yang sesuai berdasarkan kata-kata atau frasa-frasa kunci tersebut. Misal jika *user* memberikan kata kunci “berak” dan kata kunci “darah” dengan hasil terjemahan X untuk proses pembelajaran berdasarkan kata-kata atau frasa-frasa kunci, maka jika nanti *user* memasukkan *input* “berak darah” atau “berak mengeluarkan darah”, aplikasi akan menampilkan hasil terjemahan X karena kedua *input* tersebut mengandung kata kunci “berak” dan kata kunci “darah”.

Pada pembelajaran untuk penerjemahan berdasarkan statistik, *user* memasukkan contoh masukan keluhan beserta hasil terjemahannya ke dalam *database* aplikasi, kemudian aplikasi akan mengecek apakah contoh masukan keluhan tersebut mirip dengan salah satu entri yang sudah dimasukkan pada *database* sebelumnya atau tidak, jika tidak ada yang mirip maka aplikasi akan memasukkan contoh masukan keluhan tersebut beserta hasil terjemahannya sebagai entri baru, namun jika sudah ada entri yang mirip maka entri tersebut akan diperbaharui data statistiknya yaitu data probabilitas entri tersebut

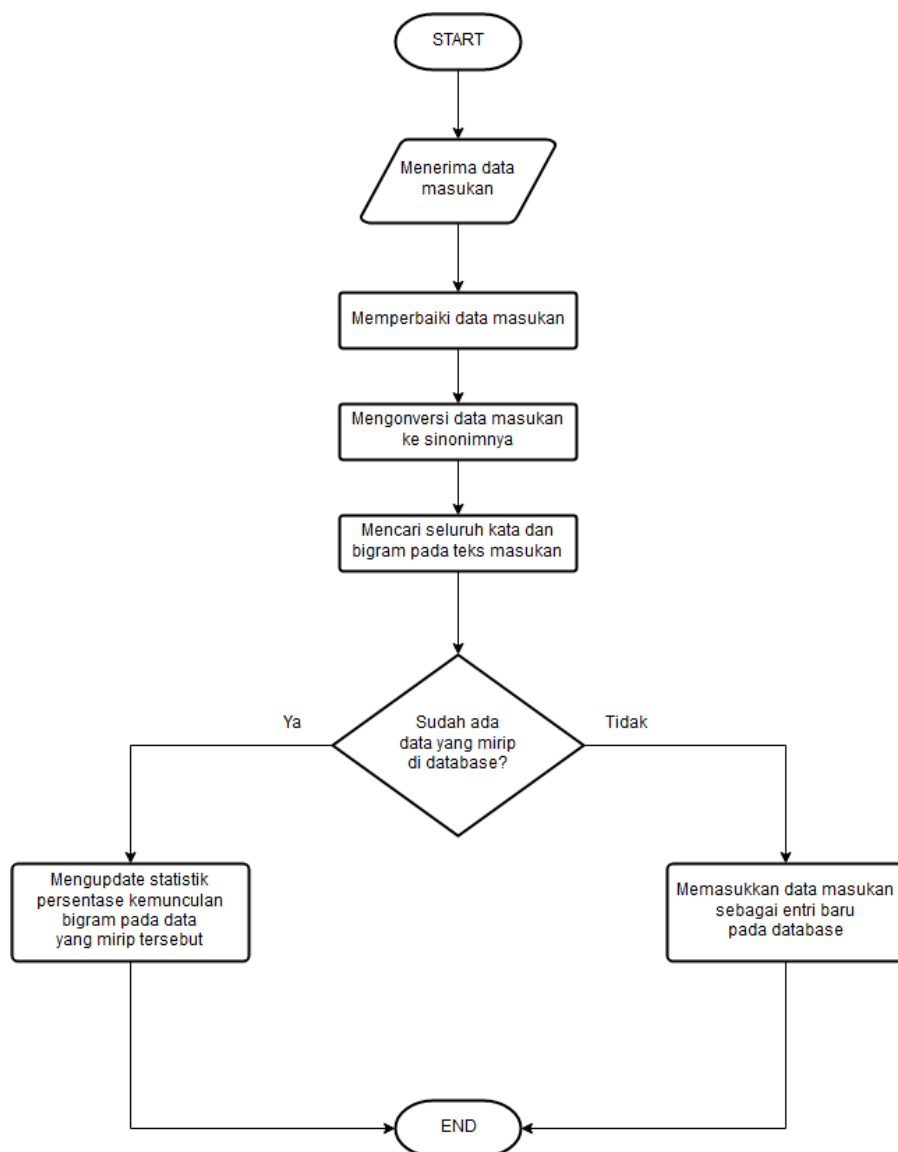
mengandung bigram tertentu. Pengecekan kemiripan adalah berdasarkan jumlah kata, kata-katanya, dan bigram-bigramnya. Dua kata yang memiliki makna yang sama dianggap sebagai kata yang sama. Untuk bisa disebut mirip, contoh masukan keluhan harus berisi jumlah kata dan kata-kata yang sama dengan salah satu entri yang ada pada *database*, serta memiliki jumlah bigram yang sama minimal sebanyak jumlah kata dibagi dua. Nantinya jika *user* memasukkan keluhan maka aplikasi akan mencari apakah keluhan tersebut mirip dengan salah satu entri yang ada pada *database* aplikasi atau tidak. Masukan keluhan dianggap mirip dengan salah satu entri pada *database* jika memiliki jumlah kata yang sama, memiliki kata-kata yang sama, dan memiliki bigram yang sama minimal sebanyak jumlah kata dibagi dua. Jika ada dua atau lebih entri yang mirip, maka dipilih entri yang paling mirip bigram-bigram nya sebagai entri yang paling mirip dengan masukan keluhan, jika masih ada dua atau lebih entri yang mirip maka akan dipilih entri yang pertama ditemukan. Jika ditemukan entri yang mirip maka aplikasi akan menampilkan terjemahan yang sesuai berdasarkan entri tersebut. Dengan menggunakan metode pembelajaran ini, jika *user* memberikan contoh masukan “sakit telinga kemarin” dengan hasil terjemahan X untuk proses pembelajaran berdasarkan statistik, maka jika nanti *user* memasukkan *input* “sakit kuping kemarin” atau “kemarin sakit telinga”, aplikasi akan menampilkan hasil terjemahan X karena kedua *input* tersebut memiliki jumlah kata yang sama, kata-kata yang sama, serta minimal memiliki jumlah bigram sama sebanyak jumlah kata dibagi dua, dengan entri yang sudah dimasukkan ke dalam *database*.

g. Menampilkan Hasil Penerjemahan

Setelah proses penerjemahan selesai, hasil terjemahan akan ditampilkan kepada *user* yang isinya antara lain nama umum keluhan, lokasi keluhan, sistem organ keluhan, istilah medis keluhan, awal kemunculan keluhan, frekuensi keluhan, dan sifat keluhan. Sebelumnya teks hasil terjemahan akan diperbaiki dulu agar sesuai dengan tampilan keluaran yang diinginkan. Hasil terjemahan ini juga dapat didiktekan oleh aplikasi menjadi suara yang dapat didengar oleh *user* dengan menggunakan fitur *speech synthesis* pada aplikasi. Teks hasil terjemahan ini dapat disimpan ke dalam file berformat teks atau excel, dan suara yang dihasilkan oleh fitur *speech synthesis* dapat disimpan ke dalam *file* audio berformat WAV. Hasil terjemahan ini nantinya dapat digunakan juga untuk masukan fitur bantuan diagnosis pada aplikasi yang akan menampilkan kemungkinan penyakit yang diderita oleh pasien beserta rekomendasi pengobatannya jika ada. Fitur bantuan diagnosis ini adalah fitur tambahan yang datanya nantinya dapat diisi sendiri oleh dokter.

4.2 Algoritma Pembelajaran untuk Penerjemahan

Aplikasi yang dikembangkan dapat mempelajari keluhan baru dengan data pembelajaran yang diberikan oleh *user*. Untuk pembelajaran menggunakan contoh dan *keyword*, algoritmanya pembelajarannya cukup sederhana, yaitu masukan oleh *user* diperbaiki dulu isinya oleh aplikasi, diubah ke sinonimnya, dan kemudian langsung dimasukkan ke dalam *database*. Untuk pembelajaran menggunakan data statistik, algoritma pembelajarannya dapat dilihat pada *flowchart* pada gambar 4.2. di bawah ini:



Gambar 4.2. Algoritma pembelajaran dengan data statistik.

Untuk pembelajaran menggunakan data statistik, pertama-tama *user* memasukkan data masukan lalu kemudian aplikasi memperbaiki data masukan tersebut seperti mengubah semua katanya menjadi huruf kecil semua, menghilangkan spasi ganda, memperbaiki penulisan tanda baca yang kurang tepat, menghapus spasi di awal dan di akhir teks masukan (*trimming*), dan memperbaiki singkatan-singkatan yang tidak baku. Setelah data masukan diperbaiki, isi teks pada data masukan tersebut kemudian dikonversi ke sinonimnya untuk mencegah kata yang sama dianggap berbeda oleh aplikasi. Kemudian aplikasi mencari seluruh kata dan bigram yang ada di dalam teks contoh keluhan pada data masukan.

Aplikasi kemudian mengecek apakah sudah ada data yang mirip di dalam *database*. Pengecekan kemiripan ini dilakukan dengan melakukan pengecekan sebagai berikut:

1. Apakah teks contoh keluhan pada data sebelumnya memiliki jumlah kata yang sama dengan jumlah kata pada teks contoh keluhan pada data yang dimasukkan?
2. Apakah teks contoh keluhan pada data sebelumnya memiliki seluruh kata-kata yang terdapat pada teks contoh keluhan pada data yang dimasukkan tanpa mempedulikan urutannya?
3. Apakah teks contoh keluhan pada data sebelumnya memiliki bigram yang mirip sejumlah setengah dari banyak kata yang terdapat pada teks contoh keluhan pada data yang dimasukkan?
4. Apakah data lokasi, sistem organ, dan istilah medis pada data sebelumnya sama dengan data yang ada pada data masukan.

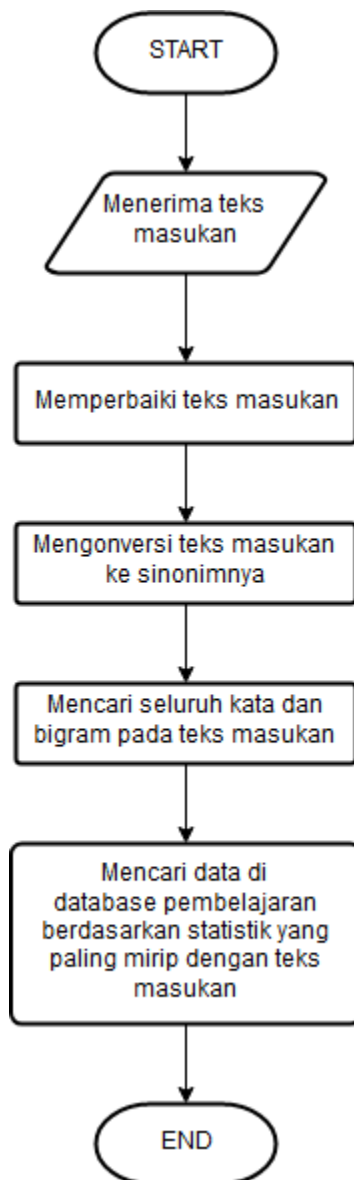
Jika jawaban dari hasil pengecekan keempat pertanyaan tersebut adalah “ya”, maka data masukan digunakan untuk mengubah data statistik probabilitas kemunculan bigram pada data yang mirip yang sudah ada di *database*. Namun jika ada satu saja jawaban tidak dari hasil pengecekan keempat pertanyaan di atas, maka data masukan dimasukkan sebagai entri baru pada *database*.

4.3. Algoritma Penerjemahan Menggunakan Pembelajaran Berdasarkan Statistik

Untuk penerjemahan menggunakan hasil pembelajaran berdasarkan statistik, pertama-tama *user* memasukkan *input* ke dalam aplikasi, kemudian aplikasi memperbaiki *input* tersebut yaitu dengan mengubah semua katanya menjadi huruf kecil semua, menghilangkan spasi ganda, memperbaiki penulisan tanda baca yang kurang tepat, menghapus spasi di awal dan di akhir teks masukan (*trimming*), dan memperbaiki singkatan-singkatan yang tidak baku. Teks masukan kemudian diubah ke sinonimnya untuk mencegah aplikasi menganggap kata yang sama artinya sebagai kata berbeda. Kemudian aplikasi mencari data di dalam *database* yang paling mirip dengan teks masukan dengan melakukan pengecekan sebagai berikut:

1. Apakah teks contoh keluhan pada data sebelumnya memiliki jumlah kata yang sama dengan jumlah kata pada teks contoh keluhan pada data yang dimasukkan?
2. Apakah teks contoh keluhan pada data sebelumnya memiliki seluruh kata-kata yang terdapat pada teks contoh keluhan pada data yang dimasukkan tanpa mempedulikan urutannya?
3. Apakah teks contoh keluhan pada data sebelumnya memiliki bigram yang mirip sejumlah setengah dari banyak kata yang terdapat pada teks contoh keluhan pada data yang dimasukkan?

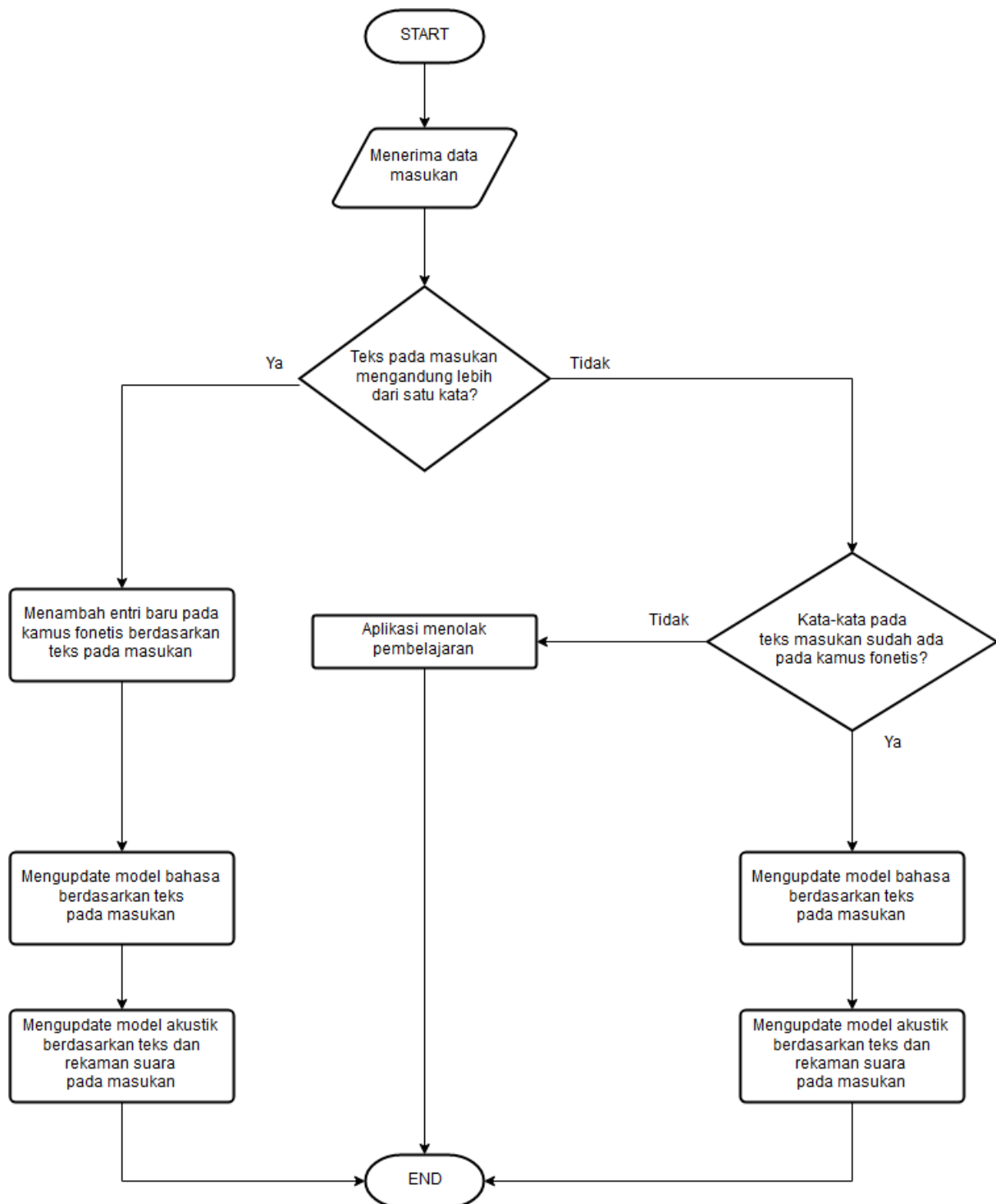
Data yang sesuai adalah data yang memberikan jawaban “ya” pada ketiga pertanyaan di atas. Jika hanya ada satu data yang sesuai pada data di *database*, maka data tersebut yang digunakan untuk penerjemahan. Jika ada lebih dari satu data yang sesuai, pilih data yang memiliki probabilitas bigram paling mendekati data masukan. Jika tidak ada data yang sesuai maka proses penerjemahan gagal dilakukan. Algoritma penerjemahan menggunakan data pembelajaran statistik dapat dilihat *flowchart* nya pada gambar 4.3. di bawah ini:



Gambar 4.3. Algoritma penerjemahan menggunakan hasil pembelajaran berdasarkan statistik.

4.4 Algoritma Pembelajaran untuk *Speech recognition*

Pembelajaran aplikasi untuk *speech recognition* dilakukan dengan memasukkan *input* contoh suara dan teks transkripsi yang sesuai dari contoh suara tersebut. *Flowchart* algoritma pembelajaran untuk *speech recognition* dapat dilihat pada gambar 4.4. di bawah ini.



Gambar 4.4. Algoritma pembelajaran untuk *speech recognition*.

Penjelasan dari *flowchart* di atas adalah sebagai berikut. Setelah *user* memasukkan *input* teks dan suara, aplikasi memeriksa apakah *input* teks tersebut hanya berisi satu kata atau lebih. Jika hanya berisi satu kata, maka data teks masukan digunakan untuk meng-

update isi kamus fonetis dan model bahasa, sedangkan data masukan suara dan teks digunakan untuk meng-*update* model akustik. Jika berisi lebih dari satu kata dicek lagi apakah kata-kata tersebut sudah ada di dalam kamus fonetis atau belum, jika belum maka aplikasi menolak proses pembelajaran, jika sudah maka data masukan teks digunakan untuk meng-*update* isi model bahasa dan data masukan teks dan suara digunakan untuk meng-*update* model akustik.