APLIKASI PREDICTIVE TEXT BERBAHASA INDONESIA DENGAN METODE N-GRAM

Silvia Rostianingsih [1)], Sendy Andrian Sugianto[2)], Liliana[3)]
1, 2, 3) Program Studi Teknik Informatika Fakultas Teknologi Industri Universitas Kristen Petra
1, 3) E-mail: silvia@petra.ac.id, lilian@petra.ac.id

Abstrak

Aplikasi untuk memproses kata dengan dibantu prediksi kemunculan sebuah kata membantu mempercepat proses pengetikkan. *Predictive text* banyak diaplikasikan pada perangkat komunikatif yang membutuhkan input text, seperti komputer, personal digital assistant (PDA), dan telepon selular. Metode lain yang dapat digunakan adalah n-gram. Metode n-gram digabung dengan fungsi *scoring* yang mendukung prediksi kata, yaitu language model dan *semantic affinity*. *Language model* didasarkan pada urutan kata dan kata yang paling sering digunakan dalam *input*-an teks, sedangkan *semantic affinity* didasarkan pada kemungkinan kata-kata tersebut muncul bersama dalam sebuah kalimat. Proses dimulai dengan memecah kata per kata dan mengelompokkannya sesuai dengan language model. Selanjutnya dilakukan proses *scoring* untuk menentukan kata mana yang sesuai untuk menjadi pilihan prediksi kata. Hasil pengujian menunjukkan bahwa metode bigram dan trigram dalam language model mempengaruhi sistem *predictive text*, karena setiap scoring yang dilakukan oleh sistem mengacu pada *language model* dari kata yang diprediksi. *Keystroke saving* yang dihasilkan dapat mencapai angka 25% bergantung pada data training. Rata-rata prediksi efektif terjadi di atas 30% dari total prediksi yang terjadi. Hal ini dikarenakan oleh pengaruh dari language model yang dapat memprediksi kata dengan lebih efektif dan akurat.

Kata kunci : *N-gram, Keystroke Saving, Predictive Text*

Pendahuluan

Untuk dapat membantu pengguna aplikasi editor text agar dapat mengetikkan nasikahnya lebih cepat, maka salah satunya adalah diperlukannya fungsi yang dapat memprediksi kata-kata apa akan digunakan oleh pengguna berdasarkan kalimat yang sering diketikkan. Sehingga dengan hanya mengetikkan beberapa karakter awal dari kata tersebut, prediksi kata-kata yang dimaksud dapat muncul dengan sendirinya.

Pemrograman dilakukan menggunakan bahasa C# dan penyimpanan data menggunakan file text. Aplikasi yang dibuat adalah editor yang dapat menghasilkan file .txt. Cara pengujian yang dilakukan adalah dengan menginputkan beberapa kalimat bahasa Indonesia pada aplikasi editor sebagai isi file, lalu disimpan terlebih dahulu. Setelah itu, dibuka halaman baru untuk mengetikkan kembali kalimat yang mirip dengan yang telah diinputkan untuk menilai seberapa akurat *predictive text* yang muncul sesuai dengan kalimat awal yang sudah diinputkan

Studi Pustaka

Predictive Text adalah sebuah fitur pada pengetikan yang bertujuan mengurangi *keystroke* dalam pengetikan dengan cara memprediksi kata yang akan muncul berdasarkan huruf yang diketikkan. Metode T9 adalah metode yang digunakan pada *mobile device* yaitu membandingkan kemungkinan kata ke *database* linguistik untuk "menebak" kata dimaksudkan (M. Silfverberg , 2000)]. Metode lain adalah n-gram yang dilakukan dengan menghitung probabilitas bersyarat untuk sebuah kata dari urutan kata sebelumnya (Yousef Bassil, 2012). N-gram tidak terlalu sensitif terhadap kesalahan penulisan yang terdapat pada suatu dokumen (Ahmad Hanafi, 2009). Sebuah *n-gram* adalah sebuah kumpulan kata dengan masing-masing memiliki panjang n kata. Sebagai contoh, sebuah n-gram ukuran 1 disebut sebagai unigram; ukuran 2 sebagai “bigram”; ukuran 3 sebagai "trigram", dan seterusnya.

Untuk mendukung *predictive text* maka dilakukan penggabungan metode n-gram dengan fungsi *scoring* seperti *language model* dan *semantic affinity*. *Language model* didasarkan pada urutan kata dan kata yang paling sering digunakan dalam *input*-an teks, sedangkan *semantic affinity* didasarkan pada kemungkinan kata tersebut muncul bersama dalam sebuah kalimat (Yousef Bassil, 2012). *Language Model* adalah metode *input* berdasarkan frekuensi kata dan tidak peka terhadap

konteks dan bobot ditentukan dengan mengiterasi subset dari kombinasi yang mungkin (Jakob Jorwall, 2009). Sebuah interpolasi linier dari *n-gram* yang berbeda menjadi:

$$P(w_t|w_{t-n+1}..w_{t-1}) = \lambda_t P(w_t) + ...+ \lambda_n P(w_t|w_{t-n+1}..w_{t-1}) \quad (1)$$

λ_i = *weights* t untuk spesifik *n-gram* $0 \leq \lambda_i \leq 1$

Dengan menggunakan *n-gram* yang panjangnya tidak lebih dari tiga, model hanya membutuhkan kata-kata tersebut berada dalam sebuah kelompok yang masing-masing maksimal hanya berisi tiga kata di dalam korpus. *Semantic affinity* antara dua kata untuk mengukur hubungan dalam kalimat (Jianhua Li and Graeme Hirst, 2005), terdefinisi sebagai berikut:

$$SemR(w_i, w_j) = \frac{C(w_i, w_j)}{C(w_i)C(w_j)} \quad (2)$$

$C(w_i,w_j)$ = jumlah berapa kali kata w_i dan w_j terjadi dalam sebuah kalimat di dalam korpus
$C(w_i)$ = jumlah kata w_i di dalam korpus.

Hubungan simetrisnya adalah :

$$C(w_i, w_j) = C(w_j, w_i) \quad (3)$$

Afinitas semantiknya diperkirakan dari sebuah kata w, didefinisikan sebagai berikut:

$$SemA(w|H) = \sum_{w_i \in H} SemR(w, w_i) \quad (4)$$

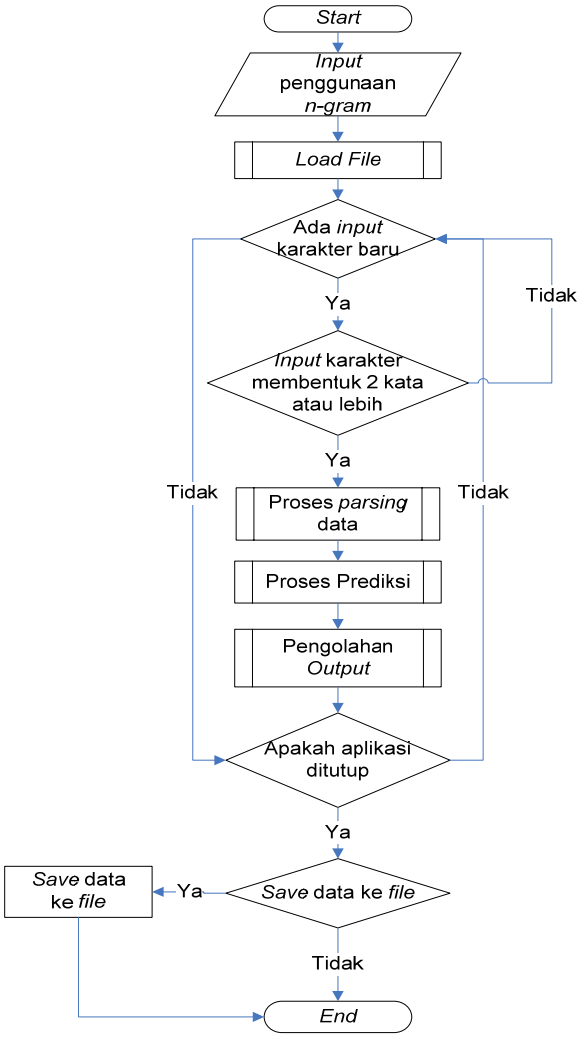
H = konteks kata w (mengandung kata-kata dari sebelah kiri kata saat ini)

Metodologi Penelitian

Metodologi penelitian dilakukan dengan mempelajari tentang metode N-Gram-Based, dilanjutkan dengan membuat perancangan sistem tentang pengolahan kata dan metode scoring kata. Pembuatan perangkat lunak yaitu dengan mengimplementasikan desain sistem yang telah dibuat ke dalam bahasa pemrograman, meliputi language model, frequency scoring, semantic scoring, Ngram scoring. Selanjutnya dilakukan pengujian aplikasi dalam melakukan prediksi dan *keystroke saving* yang dihasilkan oleh tiap metode *scoring*. Kesimpulan dilakukan dengan membandingkan hasil prediksi dan *keystroke saving* yang dihasilkan dari aplikasi.

Desain Sistem

Dalam melakukan *predictive text*, user terlebih dahulu memasukkan metode *n-gram* yang digunakan. Selanjutnya sistem akan melakukan *load file* kata yang ada sesuai metode *n-gram* yang dipilih. Sistem akan membaca *input* karakter dari *user* dan melakukan *parsing* data. Selanjutnya sistem melakukan *searching* dan *scoring* kata dari *file* untuk menghasilkan *predictive text*. Terakhir, sistem memberikan usulan kata yang menjadi *predictive text* kepada *user*. Rancangan sistem kerja aplikasi secara garis besar ditunjukkan pada Gambar 1.



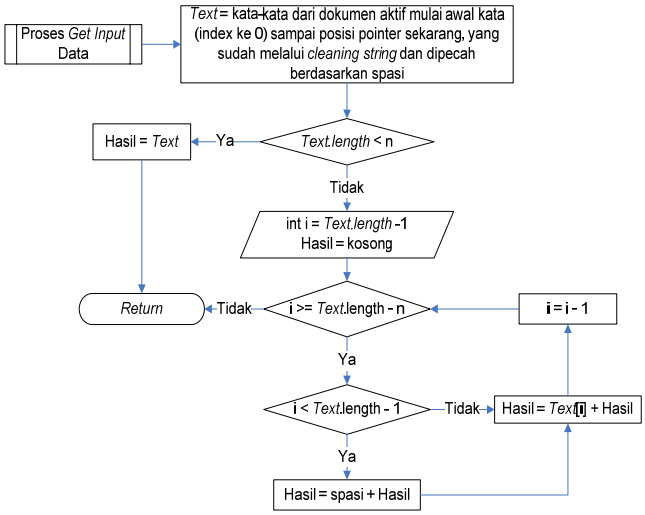
Gambar 1. Desain Sistem Kerja *Predictive Text*

Parsing dilakukan saat ada input huruf dari *user*. *Parsing* dilakukan jika pada aplikasi telah tersimpan 2 kata atau lebih. *Parsing* adalah proses untuk membagi kata-kata menjadi rangkaian kata sepanjang n, dimana n adalah sesuai metode n-gram yang dipilih. Untuk proses *parsing* data sendiri, kata-kata yang di-*parsing* adalah kata-kata dari dokumen baru / dokumen yang sedang aktif yang telah terdapat *input*-an karakter baru dari *user* dan sudah membentuk kata.

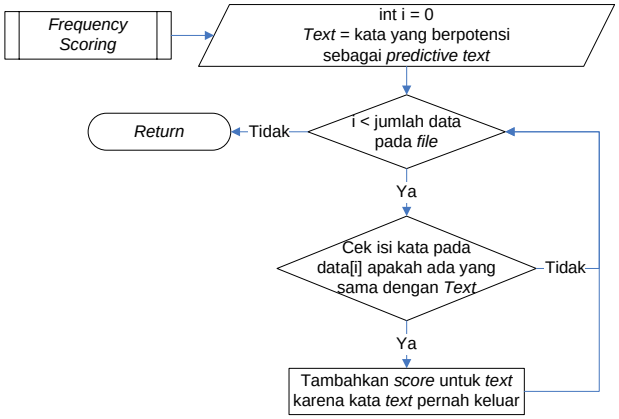
Proses *cleaning string* dilakukan untuk membersihkan kata dari tanda baca atau lain yang tidak dibutuhkan. Kata dijadikan *lower case*, selanjutnya dipecah berdasarkan spasi menjadi kumpulan kata yang terdiri dari satu kata. Selanjutnya kata dikelompokkan sesuai dengan metode *n-gram* yang dipilih.

Proses *Get Input Data* pada Gambar 2 digunakan untuk menentukan kata yang diusulkan menjadi pilihan kata *predictive text*. Pertama dilakukan pencarian kata yang dapat diusulkan menjadi pilihan kata *predictive text* dari *file* (proses *searching*), dan yang kedua adalah proses penilaian dari tiap kata yang telah dipilih untuk menjadi pilihan *predictive text* (proses *scoring*).

Setelah mendapat *input*, proses berikutnya adalah *frequency scoring* pada Gambar 3 yaitu mengolah data input untuk mencari kata yang berpotensi menjadi usulan kata *predictive text* pada *file*. Usulan kata diolah untuk menjadi pilihan *predictive text*. Ada 3 penilaian yang dilakukan, yang pertama adalah penilaian berdasarkan frekuensi kata pernah keluar, baik dalam dokumen yang sedang diketik *user* maupun dalam dokumen-dokumen sebelumnya.

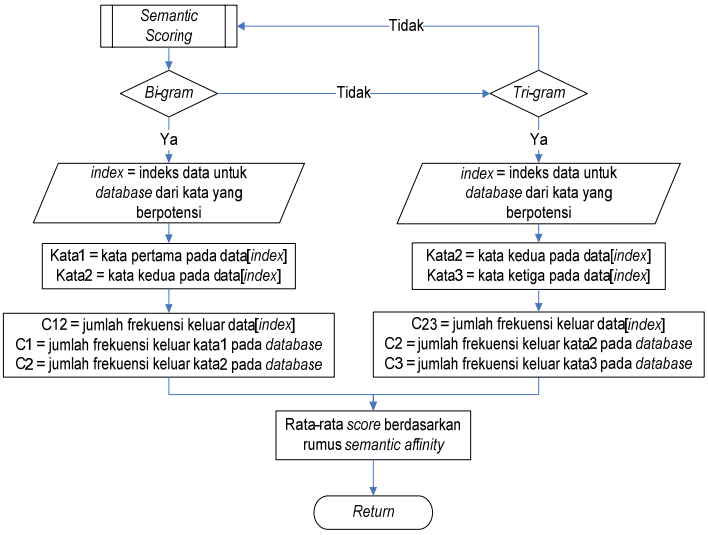


Gambar 2. Diagram Alur Proses *Get Input Data*



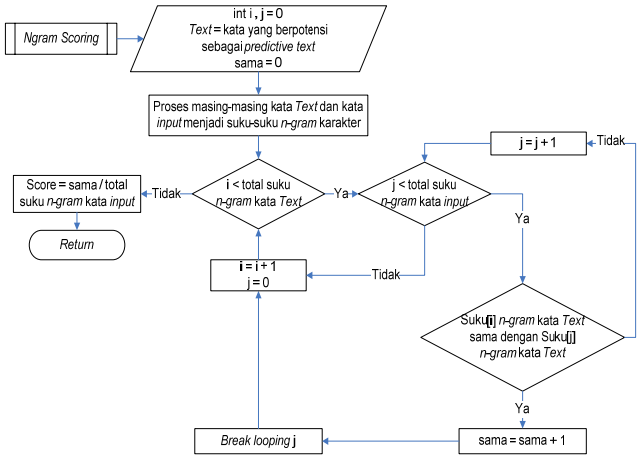
Gambar 3. Diagram Alur Proses Frequency Scoring

Penilaian berdasarkan hubungan semantik pada Gambar 4 dilakukan berdasarkan hubungan semantik kata yang pernah terjadi. Hubungan semantik kata adalah hubungan antara kata yang satu dengan lain (kata tersebut pernah muncul dalam dokumen yang disimpan sebelumnya).



Gambar 4. Diagram Alur Proses Semantic Scoring

Proses terakhir adalah n-gram *scoring* pada Gambar 5 yaitu menilai berdasarkan kesamaan suku *n-gram* yang dimiliki oleh kata kandidat dibandingkan dengan suku *n-gram* dari kata *input.*



Gambar 5. Diagram Alur Proses Ngram Scoring

Hasil dan Pembahasan

Pengujian dilakukan antara lain menguji bobot dari tiap metode *scoring*-nya, yakni *Keystroke Saving* (KS) dan *Score* Prediksi Efektif (SPE). Pengujian dengan menghitung *keystroke saving* adalah untuk menghitung seberapa banyak karakter yang dapat dihemat untuk menghasilkan sebuah teks tertentu. SPE didapat dari jumlah prediksi efektif yang terjadi dibandingkan dengan jumlah total prediksi yang terjadi. Hasil perhitungan yang didapat dari pengujian pada Bigram (Tabel 1) dan Trigram (Tabel 2) menunjukkan nilai yang hampir sama. Sedangkan untuk persentase frekuensi kata keluar (Tabel 3) menunjukkan bahwa metode bigram dan trigram masih memiliki persentase frekuensi kata keluar yang lebih tinggi dibanding metode lainnya.

Tabel 1. Hasil Perhitungan Formula Pada Bigram

No	Nama Data	Total Kata	Total Prediksi	Prediksi Efektif	KS (%)	SPE (%)
1	Artikel Opini Ekonomi Politik	726	466	220	25,47	47,21
2	Berita Honda	201	97	41	17,16	42,27
3	Artikel Trio Harry Potter	135	57	19	15,17	33,33
4	Artikel Usir Ketombe	178	87	36	19,36	41,38
5	Berita *Gadget* Mozilla	182	85	31	16,92	36,47
6	Berita Pembangunan Rusun	135	65	32	17,36	49,23
7	Artikel Opini untuk Siswa	633	345	137	20,5	39,71

Tabel 2. Hasil Perhitungan Formula Pada Trigram

No	Nama Data	Total Kata	Total Prediksi	Prediksi Efektif	KS (%)	SPE (%)
1	Artikel Opini Ekonomi Politik	726	462	203	25,07	43,94
2	Berita Honda	201	93	39	17,04	41,94
3	Artikel Trio Harry Potter	135	58	20	15,06	34,48
4	Artikel Usir Ketombe	178	87	38	20,03	43,68
5	Berita *Gadget* Mozilla	182	84	32	16,77	38,10

6	Berita Pembangunan Rusun	135	64	32	17,16	50,00
7	Artikel Opini untuk Siswa	633	339	130	20,09	38,35

Tabel 3. Persentase Frekuensi Kata Keluar

Banyak	Presentase Frekuensi Kata Keluar			
	2-gram	3-gram	4-gram	5-gram
>8x	0,03%	0,00%	0,00%	0,00%
6-8x	0,04%	0,03%	0,00%	0,00%
3-5x	0,36%	0,21%	0,03%	0,00%
2x	0,94%	2,60%	0,65%	0,52%
1x	98,63%	97,16%	99,32%	99,48%

Kesimpulan

Dari hasil penelitian dapat disimpulkan:

a. Rata-rata *keystroke saving* yang dihasilkan pada pengujian ini adalah 15 hingga 25 persen bergantung pada data *training*.
b. Rata-rata prediksi efektif terjadi di atas 30% dari total prediksi yang terjadi. Hal ini dikarenakan oleh pengaruh dari *language model* yang dapat langsung memprediksi kata dengan lebih efektif dan akurat.
c. Frekuensi dari language model yang tinggi sangat mempengaruhi scoring sistem, karena semakin tinggi frekuensi language model suatu kata, maka akan semakin tinggi pula bobot / nilai dari kata itu sendiri.
d. Semakin besar nilai n dalam *n-gram* berbanding terbalik dengan jumlah frekuensi keluar yang didapat, yaitu semakin kecil atau lebih jarang keluar. Penggunaan model *bi-gram* dan *tri-gram* untuk *language model* masih memungkinkan, karena hasil dari jumlah frekuensi keluar pada suku *n-gram*-nya masih cukup besar dan datanya masih valid apabila diproses lebih lanjut.

Daftar pustaka

Ahmad Hanafi, 2009, *Pengenalan Bahasa Suku Bangsa Indonesia Berbasis Teks Menggunakan Metode N-gram*. Tugas Akhir IT TELKOM.

Jakob Jorwall and Sebastian Ganslandt, 2009, *Context-Aware Predictive Text Entry for Swedish using Semantics and Syntax*. Retrieved January 12, 2013, from http://fileadmin.cs.lth.se/cs/Personal/Pierre_Nugues/memoires/sebastian_jakob /sebastian_jakob.pdf.

Jianhua Li and Graeme Hirst, 2005, *Semantic knowledge in word completion, In Assets '05: Proceedings of the 7th international ACM SIGACCESS conference on Computers and accessibility*, Baltimore.

M. Silfverberg, I.S. MacKenzie, and P. Korhonen, 2000, Predicting Text Entry Speed on mobile phones, *Proceedings of the ACM Conference on Human Factors in Computing Systems - CHI 2000*, pp. 9-16, New York: ACM.

Yousef Bassil, 2012, Parallel Spell Checking Algorithm Based on Yahoo! N-Grams Dataset, *International Journal of Research and Reviews in Computer Science (IJRRCS), Vol. 3, No. 1, February 2012*, Lebanese Association for Computational Sciences.