BAB VI
PENUTUP

6.1. Kesimpulan

Dari hasil penelitian yang telah dilakukan dapat diambil kesimpulan sebagai berikut.

1. Penggunaan algoritma Jaro-Winkler *distance* hanya dapat melakukan pengecekan kata secara langsung berdasarkan *dataset* yang ada. Untuk melakukan pengecekan kata-kata entitas dibutuhkan penambahan model untuk *Name Entity Recognation* (NER).
2. Penggunaan algoritma Jaro-Winkler *distance* pada penelitian ini belum dapat menentukan nilai akurasi model secara langsung.
3. Waktu eksekusi model tergantung dari jumlah *dataset* dan data masukan yang digunakan. Semakin banyak data semakin lama pula waktu eksekusinya. Waktu eksekusi dapat dipercepat menggunakan *parallel processing. Parallel processing* juga tidak menjamin waktu eksekusi dapat meningkat dengan signifikan tergantung dari spesifikasi komputer yang digunakan.

6.2. Saran

Berdasarkan penelitian yang sudah dilakukan masih banyak yang harus dikembangkan seperti pengecekan untuk kata-kata entitas atau pengecekan secara gramatikal serta dapat mengecek dokumen sesuai dengan aturan Bahasa Indonesia yang baku. Pengembangan pengecekan kata-kata entitas dapat dilakukan dengan mengimplementasikan model *Named Enitiy Recognition* (NER). Selain itu diperlukan juga pengembangan untuk pengecekan nilai akurasi dari model ini. Dari sisi aplikasi perlu juga ditambahkan agar aplikasi dapat menampilkan teks dari dokumen sesuai dengan format teks asli nya misalnya bentuk paragrafnya.

Daftar Pustaka

[1] A. I. Fahma, I. Cholissodin, and R. S. Perdana, "Identifikasi Kesalahan Penulisan Kata (Typographical Error) pada Dokumen Berbahasa Indonesia Menggunakan Metode N-gram dan Levenshtein Distance," *J. Pengemb. Teknol. Inf. dan Ilmu Komput.*, vol. 2, no. 1, pp. 53–62, 2018.

[2] M. S.- Pgsd and S. Nuuwar, "Kesalahan Berbahasa, Penulisan Kata Baku," pp. 74–82, 2009.

[3] L. T. Hue and H. A. Jalil, "International Journal of Instruction," *Int. J. Instr.*, vol. 6, no. 2, pp. 53–66, 2013.

[4] W. W. A. Umboh, S. R. Sentinuwo, and A. M. Sambul, "Rancang Bangun Aplikasi Deteksi Kesalahan Penulisan Naskah Dokumen Skripsi," *E-Journal Tek. Inform.*, vol. 11, no. 1, 2017.

[5] Y. Rochmawati and R. Kusumaningrum, "Studi Perbandingan Algoritma Pencarian String dalam Metode Approximate String Matching untuk Identifikasi Kesalahan Pengetikan Teks," *J. Buana Inform.*, vol. 7, no. 2, pp. 125–134, 2016.

[6] C. I. Ratnasari, S. Kusumadewi, and L. Rosita, "A Non-Word Error Spell Checker for Patient Complaints in Bahasa Indonesia," *Int. J. Inf. Technol. Comput. Sci. Open Source*, vol. 1, no. 1, pp. 18–21, 2017.

[7] P. H. Hema and C. Sunitha, "Spell Checker for Non Word Error Detection : Survey," *Int. J. Adv. Res. Comput. Sci. Softw. Eng.*, vol. 5, no. 3, pp. 360–363, 2015.

[8] G. G. Chowdhury, "Natural language processing," *Annu. Rev. Inf. Sci. Technol.*, vol. 106, pp. 213–226, 2016.

[9] J. Pustejovsky and a Stubbs, *Natural language annotation for machine learning*. 2013.

[10] P. A. V Hall, "Approximate String Matching," no. December, 1980.

[11] W. E. Winkler and U. S. Bureau, “The State of Record Linkage and Current Research Problems,” 1962.

[12] Ilya Ilyankou, “IB Extended Essay Computer Science Comparison of Jaro-Winkler and Ratcliff / Obershelp algorithms in spell check,” no. May, pp. 1–35, 2014.

[13] M. Kamayani, M. Y. Soleh, and A. Purwarianti, “Application of Document Spelling Checker for Bahasa Indonesia,” no. January, 2011.

[14] R. Gadde and J. Kelly, “Using Multiprocessing to Make Python Code Faster,” *Medium - Data@Urban*, 2018. [Online]. Available: https://medium.com/@urban_institute/using-multiprocessing-to-make-python-code-faster-23ea5ef996ba#8597. [Accessed: 28-Jul-2019].