**Chapter 14 : Counting Words**

Dijelaskan bahwa sejahu pemahaman yang telah didapatkan dari bab 2 sampai 7. Menegaskan bahasa memiliki struktur yang luar biasa, sebagian terlihat, dan sebagian tersembunyi, dan masih banyak yang harus ditemukan. Dijelaskan oleh penulis bahwa sampai dengan buku ini diterbitkan penelitian yang dilakukan adalah qualitative dimana merujuk kepada pemaknaan yang muncul dari fonem atau grafem(Tag bagian dari ucapan, milik konstituen, dll., adalah bagian dari informasi kualitatif yang dapat kita kumpulkan pada morfem, kata, atau kelompok kata,) dilakukan dengan melakukan tanya jawab.

Terhadap pendekatan kuantitatif, dilakukan dalam tahapan dan langkah yang berbeda. menyimpulkan dengan melampirkan informasi numerik ke unit bahasa. Lalu diberikan beberapa contoh proses perhitungan informasi numerik daei unit bahasa dari saman ahli ahli taurat Yahudi yang disebut Masoret sampai kepada Matematikawan Rusia (1913) A Makrov mengambil 20.000 grafem dari sebuah novel. Selanjutnya penulis berkesimpulan bahwa komputer pun dapat melakukan tugas menghitung dengan baik. Sehingga n inilah sebabnya pendekatan statistik terhadap bahasa mengalami peningkatan luar biasa pada masa-masa awal komputasi dan peningkatan besar kedua pada tahun sembilan puluhan ketika komputer pribadi dan peningkatan daya pemrosesan membuat metode statistik yang rumit dapat diakses oleh setiap orang.

1. **Tokenization dan Segmentasi**

Tokenisasi adalah proses memecah teks menjadi unit-unit yang lebih kecil yang disebut token, yang bisa berupa kata, tanda baca, atau entitas lain tergantung konteks dan bahasa. Konsep “kata” sendiri sulit didefinisikan secara universal, sehingga pendekatan statistik menggunakan tokenisasi untuk menghindari definisi baku dan membuatnya bergantung pada bahasa dan tugas tertentu. Dalam buku disajikan contoh tokenisasi dan pengunaannya dalam Python. Misalkan dengan menggunakan *split() , nltk.word\_tokenize,* juga dengan *spacy.* Untuk bahasa tanpa spasi antar kata misalkan seperti cina dijelaskan dalam buku bahwa menghadirkan tantanagn tersendiri. Sehingg segmentasi menjadi sanggat penting. Segmentasi sendiri merupakan proses mendeteksi batas antar token dalam teks. Segmentasi adalah bagian dari proses tokenisasi.

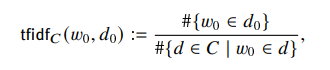
1. **Aturan Zipf’s**

Hukum Zipfs menyatakan bahwa dalam korpus teks yang cukup besar, frekuensi sebuah kata berbanding terbailik dengan peringkatnya jika diurutkan dari yang paling sering digunakan. Artinya : Semakin tinggi frekuensi sebuah kata digunakan dalam bahasa, maka semakin rendah peringkat(rangkinG) KATANYA dalam daftar frekuensi dan sebalikya. Pada buku diberikan contoh percobaan pada korpus whovian . dari gambar yang ditampilkan kurva mendekati antidiagonal menunjukan kecocokan parsial dengan hukum Zipf’s . Zipf’s Law bekerja pada data yang sudah diurutkan dari frekuensi terbesar ke terkecil. Rank adalah urutan dari kata yang paling sering muncul (rank 1) ke paling jarang (rank n). Baru setelahnya diuji apakah distribusinya mendekati 

1. **Stop Words dan tfidf**

Menurut penjelasan, Stop words adalah kata kata umum yang sering muncul dalam teks tapi kurang memberikan makna kontekstual : *the, is, and, to, of*. Dalam NLP, stop words sering dihapus dari analisis karena dianggap tidak membantu dalam membedakan dokumen atau konteks. Tidak semua stop words harus dihapus. Dalam beberapa konteks atau tugas, kata-kata ini justru penting secara semantik. Contoh : kalimat terkenal "To be or not to be" dari Hamlet hanya terdiri dari stop words—menghapusnya bisa menghilangkan makna penting.

tf-idf digunakan untuk menilai seberapa penting suatu kata dalam dokumen relatif terhadap seluruh korpus. memberi bobot pada kata berdasarkan seberapa penting kata itu dalam dokumen tertentu.

yaitu Jumlah kemunculan w di d dan Jumlah dokumen yang mengandung w.

tf (term frequency): seberapa sering kata muncul dalam dokumen tertentu. Sedangkan idf (inverse document frequency): semakin banyak dokumen yang mengandung kata tersebut, semakin rendah nilai idf-nya.

Pada buku diberikan contoh kasus episode ”Heaven Sent” dari Doctor Who. Didapatkan bahwa Kata dengan nilai tf-idf tertinggi adalah : clara, hybrid, shepherd . dan untuk nilai terendah adalah er, wait, sure, good. Kelemahan metode ini adalah tidak dapat membedakan apakah sebuah kata muncul 1 atau 1000 kali dalam dokumen lain. Yakni hanya memperhatikan Jumlah dokumen bukan frekuensi total dari dokumen tersebut.

1. **Collocations**

Pada bagian ini, konsep ***Collocation*** dijelaskan sebagai urutan kata yang sering muncul bersama dalam sebuah korpus. Untuk menghitung collocations, pertama-tama kita menghitung frekuensi pasangan kata, dengan memastikan tanda baca tidak termasuk dalam token kata. Penggunaan kode dari pustaka **nltk** memungkinkan penyaringan pasangan kata yang mengandung tanda baca. Hasil awal menunjukkan banyaknya pasangan kata yang tidak signifikan, seperti "of the" dan "in the", yang lebih merupakan kata-kata gramatikal. Untuk memperbaikinya, stopwords ditambahkan ke dalam filter, menghasilkan collocations yang lebih menarik, seperti "time lord," "human race," dan "police box." Selain itu, pengukuran kekuatan collocation dilakukan dengan menggunakan probabilitas bersyarat, di mana kita membandingkan kekuatan ketergantungan antar kata dengan menggunakan rasio log-likelihood. Hasil yang lebih memuaskan didapat dengan menggunakan rasio log-likelihood, dengan collocations kuat seperti "Lethbridge Stewart," "sonic screwdriver," dan "Captain Yates" muncul. Hasil ini dapat lebih ditingkatkan lagi dengan membatasi tag POS pada kata-kata tertentu, seperti menghindari adverbia atau angka.

1. **N-Grams**

Dalam pembahasan ini, konsep N-grams digunakan untuk membangun model bahasa yang dapat memprediksi kata berikutnya dalam sebuah kalimat berdasarkan urutan kata sebelumnya. Model bahasa ini didasarkan pada probabilitas, di mana semakin sering suatu urutan kata muncul dalam sebuah korpus, semakin tinggi probabilitas yang diberikan pada urutan tersebut. N-grams menghitung probabilitas urutan kata dengan memeriksa sekuens kata yang lebih pendek, seperti unigram (1 kata), bigram (2 kata), trigram (3 kata), dan tetragram (4 kata), dan memanfaatkan frekuensi kemunculannya untuk menentukan kata berikutnya yang paling mungkin muncul. Namun, penggunaan N-grams menghadapi tantangan dalam menghitung banyaknya kombinasi urutan kata yang dapat terjadi, yang menyebabkan sparsity data, terutama untuk N-grams dengan panjang lebih dari 3 atau 4. Dalam eksperimen dengan korpus Whovian, model autocomplete dibangun dengan menggunakan bigram, trigram, dan tetragram untuk memprediksi kata yang muncul setelah urutan kata yang sudah ada. Hasilnya menunjukkan bahwa model tetragram memberikan hasil terbaik dengan tingkat akurasi mencapai 36.20% untuk pencocokan yang tepat dengan data pelatihan. Ketika model menggunakan kombinasi tetragram, trigram, dan bigram, tingkat akurasi dapat meningkat hingga 98.40%, dengan cakupan autocompletion mencapai antara 33.67% hingga 45.90% untuk kata yang sesuai dalam data uji. Evaluasi ini dilakukan dengan metode 10-fold cross-validation untuk memastikan hasil yang lebih objektif dan mengurangi bias yang muncul dari pemilihan data uji.

1. **Vektor Semantik**

Konsep vector semantics berangkat dari hipotesis distribusional yang menyatakan bahwa kata-kata yang muncul dalam konteks serupa cenderung memiliki makna yang berkaitan. Misalnya, dalam kalimat "I eat \_ for breakfast", kata-kata seperti "eggs" atau "cornflakes" adalah pengisi yang umum karena konteksnya berkaitan dengan makanan. Walau tidak bersinonim, kata-kata tersebut dianggap *RELATED* karena sering muncul dalam konteks yang sama. Untuk membedakan, semantic similarity merujuk pada kesamaan makna secara langsung (seperti “car” dan “automobile”), sedangkan *semantic relatedness* lebih luas, mencakup hubungan makna tidak langsung, seperti antara “car” dan “gasoline”, yang sering digunakan bersama tetapi tidak berada dalam kategori makna yang sama. Komputer dapat mengenali keterkaitan semantik ini dengan mengubah konteks kata menjadi vektor biner—misalnya, kata dengan tetangga tertentu direpresentasikan sebagai vektor (0,1,0,1,0). Jika dua kata memiliki tetangga yang sama, maka hasil *scalar product* (perkalian titik) dari vektor mereka akan menunjukkan derajat keterkaitannya. Karena operasi vektor ini cepat diproses komputer, pendekatan ini menjadi dasar dari \*vector semantics\*, yaitu teknik representasi makna kata dalam bentuk vektor numerik agar hubungan semantik dapat dihitung secara efisien.

1. **LSA dan ESA**

Latent Semantic Analysis (LSA) merupakan salah satu metode awal dalam menghitung representasi vektor kata. LSA memanfaatkan *term-document matrix* yang merekam kemunculan kata dalam dokumen, dengan baris mewakili kata dan kolom mewakili dokumen. Jika suatu kata muncul dalam dokumen tertentu, maka sel pada baris dan kolom tersebut diberi nilai 1. Namun, karena matriks ini bisa sangat besar, LSA menggunakan teknik *Singular Value Decomposition* (SVD) untuk mereduksi dimensinya. Matriks besar AAA diuraikan menjadi hasil perkalian tiga matriks: A=UΣV*T* di mana Σ adalah matriks diagonal yang hanya berisi nilai positif dan biasanya dipangkas agar hanya menyisakan kkk dimensi utama yang paling signifikan. Hasilnya adalah matriks berdimensi rendah A′= yang masih cukup merepresentasikan informasi penting. Dengan ini, setiap kata dipetakan ke dalam vektor yang mencerminkan keterkaitannya dengan beberapa "dokumen virtual" atau topik, yang memudahkan analisis semantik antar kata atau dokumen menggunakan jarak kosinus.

Contoh penerapan LSA dilakukan pada *Whovian corpus*, dengan tokoh-tokoh dalam serial Doctor Who sebagai kata kunci. Analisis menunjukkan bahwa Amy dan Rory memiliki kedekatan semantik tinggi (jarak 0.026), karena sering muncul bersama dalam episode, sedangkan karakter lain memiliki jarak lebih besar tergantung frekuensi dan konteks kemunculannya.

Namun, LSA memiliki kelemahan dalam hal interpretasi, karena topik-topik laten hasil dekomposisi tidak dapat secara langsung dijelaskan atau dimaknai oleh manusia. Oleh karena itu, Gabrilovich dan Markovitch mengembangkan *Explicit Semantic Analysis* (ESA) sebagai alternatif yang lebih dapat dijelaskan. ESA menggunakan *term-document matrix* yang diisi dengan nilai *tf-idf* (bukan hanya 1 atau 0), dan dokumennya berasal dari Wikipedia. Dengan ini, setiap kata direpresentasikan oleh vektor dalam ruang halaman Wikipedia, di mana setiap koordinatnya menunjukkan pentingnya kata tersebut pada halaman tertentu. Karena ESA tidak menggunakan reduksi dimensi, semua makna tetap eksplisit dan dapat ditelusuri. Bahkan, penelitian lanjutan oleh Anderka dan Stein menunjukkan bahwa ESA tetap efektif meskipun diterapkan pada korpus selain Wikipedia, seperti artikel berita Reuters, membuktikan bahwa metode ini bersifat *corpus-independent*.

1. **Skip-gram Embeddings**
2. **Teori**

Skip-gram adalah metode pembelajaran representasi kata (word embedding) yang memungkinkan model untuk mempelajari hubungan antar kata dengan mempertimbangkan konteks sekitarnya, termasuk dengan membolehkan adanya celah (gap) antar kata, seperti pada 1-skip-n-gram. Dalam pendekatan ini, setiap kata dianggap membagi kosakata menjadi dua kelas: kata-kata yang mungkin muncul di sekitarnya (konteks) dan kata-kata yang tidak. Skip-gram bekerja dengan memulai dari representasi vektor acak untuk setiap kata, yang kemudian dioptimalkan agar semakin mencerminkan hubungan kontekstual dalam korpus. Probabilitas suatu kata muncul dalam konteks kata lain dihitung menggunakan fungsi sigmoid dari hasil perkalian dot product antar vektor. Untuk melatih model, digunakan teknik *negative sampling*, yaitu dengan memasukkan beberapa kata acak yang diasumsikan tidak berada dalam konteks target, guna menyeimbangkan antara data positif dan negatif. Proses optimasi dilakukan menggunakan *stochastic gradient descent* (SGD), di mana parameter diperbarui satu per satu untuk setiap pasangan kata target dan konteks. Setelah pelatihan selesai, vektor kata yang dihasilkan digunakan sebagai representasi embedding, baik dari vektor target maupun konteks, atau gabungan keduanya. Model ini sangat efektif untuk menangkap struktur semantik dan sintaksis dalam data teks berskala besar.

1. **Analogi Kata**

Selain dapat mengukur kemiripan kata mengunakan cosine similarity, embeddings juga bisa menangkap kemiripan kata mengunakan word analogy. Hubungan seperti “singular → plural” atau “male → female” dapat direpresentasikan sebagai vektor di ruang embedding. Contohnya : vektor (man - men) menggambarkan perubahan dari plural ke singular.

Caranya adalah pertama-tama : Model Word2Vec dibuat dari korpus kecil “dr-who.txt” menggunakan pustaka Gensim. Kemudian Teks harus diproses per baris sebagai satu kalimat dan tanpa tanda baca. Model disimpan sebagai file “dr-who.model”. Pengujian dilakukan dengan operasi vektor, seperti: women + (man - men) → hasilnya diharapkan "woman". Dari pengujian didapatkan menurut buku bahwa embedding kecil tidak cukup kuat untuk menangkap semua relasi semantik.

1. **Visualisasi**

Pada buku ini, dijelaskan cara visualisasi penyematan dengan mepergunakan Tensorflow Embedding Projector dalam bentuk 3D. mengubah model Word2Vec dari Gensim ke format .tsv agar bisa dibaca oleh alat visualisasi tersebut. Proses ini membantu untuk memahami hubungan antar kata secara visual dalam ruang vektor tiga dimensi, yang bisa berguna untuk analisis semantik atau validasi hasil pelatihan model.

* Pertama tama dibutuhkan file vektor berisi koordinat vektor kata (satu vektor per baris)
* Kemudian dibutuhkan file metadata (metadata.tsv) yang berisi daftar kata (satu kata per baris, sesuai urutan dengan file vektor).
* Langkah langkahnya dimulai dengan memuat model Word2Vec dari Gensim
* Kemudian simpan model ke format teks , lalu buka file .kv PISAHKAN VEKTOR DENGAN kata :

1. baris header dilewati agar tidak dibaca sebagai data,
2. nilai koordinat disimpan dalam vectors.tsv
3. nama kata disimpan dalam metadata.tsv

* Menurut penulis, yang penting diingat adalah Baris header (jumlah vektor dan dimensi) harus dilewati agar tidak merusak struktur data.
* Angka kecil bisa ditulis dalam notasi ilmiah (contoh: 7.734467e-05), sehingga perlu ekspresi reguler yang tepat untuk menanganinya.
* Pada hasil akhir, menurut penjabaran pada buku Setelah diunggah ke Embedding Projector, pengguna dapat melihat peta 3D dari ruang vektor kata. Kata yang dipilih akan menyoroti tetangga terdekatnya, dan tampilan dapat di-*zoom*, diputar, dan digeser secara interaktif.

1. **FastText**

* Model skip-gram seperti GloVe tidak dapat mengenali kata yang tidak ada di skorpus : neologisme, kata yang langkah dan salah ejaan.
* Maka muncul Fsttext UNTUK mengatasi masalah ini dengan memecah kata menjadi trigram grafemik (atau disebutkan urutan 3 huruf)
* Setiap kata diwakili oleh vektor hasil penjumlahan vektro trigram-nya termasuk simbol batas kata seperti < dan >
* Jika kata tidak ada di korpus, FastText tetap bisa membentuk representasinya dari trigram yang dikenal.
* Misalnya, kata "Tardis" akan dipecah menjadi trigram seperti <ta, tar, ard, rdi, dis, is>.
* Analisis korpus menunjukkan bahwa 93% dari trigram bahasa Inggris telah tercakup dalam korpus Whovian. Di implementasikan di python mengunakan pustakan *gensim*
* FastText terbukti efektif dalam mengenali kata-kata baru seperti *tardislike* atau *dalekish*, serta salah ejaan seperti *dona*.
* Model ini bekerja sangat baik untuk bahasa alfabet seperti Inggris, tetapi efektivitasnya bisa berbeda untuk bahasa non-alfabet seperti Arab atau Mandarin.
* FastText memberikan hasil yang relevan bahkan untuk kata yang tidak pernah muncul di data pelatihan.