TUGAS AKHIR – KS141501

ANALISIS TOPIK DAN AUTHOR PADA JURNAL BAHASA INGGRIS DENGAN MENGGUNAKAN METODE AUTHOR-TOPIC MODELS DAN GAUSSIAN LATENT DIRICHLET ALLOCATION (GLDA)

TOPIC ANALYSIS AND AUTHOR IN THE ENGLISH JOURNAL USING THE AUTHOR-TOPIC MODELS AND GAUSSIAN LATENT DIRICHLET ALLOCATION (GLDA)

MUCHAMMAD ANDHIKA SUPRIYANTO
NRP 0521 15 4000 085

Dosen Pembimbing
Renny Pradina K., S.T, M.T., SCJP

DEPARTEMEN SISTEM INFORMASI
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2019

TUGAS AKHIR – KS141501

ANALISIS TOPIK DAN AUTHOR PADA JURNAL BAHASA INGGRIS DENGAN MENGGUNAKAN METODE AUTHOR-TOPIC MODELS DAN GAUSSIAN LATENT DIRICHLET ALLOCATION (GLDA)

MUCHAMMAD ANDHIKA SUPRIYANTO
NRP 0521 15 4000 0085

Dosen Pembimbing
Renny Pradina K., S.T, M.T., SCJP

DEPARTEMEN SISTEM INFORMASI
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2019

Halaman ini sengaja dikosongkan

TUGAS AKHIR – KS141501

TOPIC ANALYSIS AND AUTHOR IN THE ENGLISH JOURNAL USING THE AUTHOR-TOPIC MODELS AND GAUSSIAN LATENT DIRICHLET ALLOCATION (GLDA)

MUCHAMMAD ANDHIKA SUPRIYANTO
NRP 0521 15 4000 0085

Supervisor
Renny Pradina K., S.T, M.T., SCJP

DEPARTEMENT OF INFORMATION SYSTEMS
Faculty of Information Technology and Communication
Institut Teknologi Sepuluh Nopember
Surabaya 2019

Halaman ini sengaja dikosongkan

LEMBAR PENGESAHAN

ANALISIS TOPIK DAN AUTHOR PADA JURNAL BAHASA INGGRIS DENGAN MENGGUNAKAN METODE AUTHOR-TOPIC MODELS DAN GAUSSIAN LATENT DIRICHLET ALLOCATION (GLDA)

TUGAS AKHIR

Disusun Sebagai Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Departemen Sistem Informasi
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember

Oleh:

MUCHAMMAD ANDHIKA SUPRIYANTO
NRP. 0521 15 4000 0085

Surabaya, 6 Juli 2019

KEPALA
DEPARTEMEN SISTEM INFORMASI

Mahendrawati ER, ST, M.Sc, Ph.D
NIP 19761011 200604 2 001

Halaman ini sengaja dikosongkan

LEMBAR PERSETUJUAN

ANALISIS TOPIK DAN AUTHOR PADA JURNAL BAHASA INGGRIS DENGAN MENGGUNAKAN METODE AUTHOR-TOPIC MODELS DAN GAUSSIAN LATENT DIRICHLET ALLOCATION (GLDA)

Disusun Sebagai Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Departemen Sistem Informasi
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember

Oleh:

MUCHAMMAD ANDHIKA SUPRIYANTO
NRP. 0521 15 4000 0085

Disetujui Tim Penguji: Tanggal Ujian: 8 Juli 2019
Periode Wisuda: September 2019

Renny Pradina K., S.T, M.T., SCJP **(Pembimbing I)**

Faizal Johan Atletiko, S.Kom., M.T **(Penguji I)**

Nur Aini Rakhmawati S.Kom, M.Sc.Eng, Ph.D **(Penguji II)**

Halaman ini sengaja dikosongkan

ANALISIS TOPIK DAN AUTHOR PADA JURNAL BAHASA INGGRIS DENGAN MENGGUNAKAN METODE AUTHOR-TOPIC MODELS DAN GAUSSIAN LATENT DIRICHLET ALLOCATION (GLDA)

Nama Mahasiswa	**: Muchammad Andhika Supriyanto**
NRP	**: 05211540000085**
Departemen	**: Sistem Informasi FTIK-ITS**
Dosen Pembimbing	**: Renny Pradina K., S.T, M.T., SCJP**

ABSTRAK

Penggunaan *Author-Topic* Model telah banyak dilakukan pada bidang *machine learning* khususnya *natural language processing*. Metode ini pertama kali digunakan untuk mengetahui bidang penelitian dari seorang penulis dengan melihat topik dari dokumen yang telah ditulisnya. Meskipun model ini banyak digunakan, penerapannya masih memiliki banyak kekurangan. Hal yang paling menonjol adalah ketidakmampuan *Latent Dirichlet Allocation* (*LDA*) sebagai dasar dari *Author-Topic* Model untuk menangkap kata yang belum pernah terlihat dalam data training pada saat melakukan percobaan dari suatu dokumen uji dalam menentukan distribusi kata suatu topik atau biasa disebut *out of vocabulary(OOV)*. *OOV* membuat topik yang didapat dari suatu dokumen uji tidak sesuai dengan yang diinginkan. Hal tersebut dapat terjadi karena kata-kata penting yang terdapat pada dokumen uji tidak dapat ditangkap oleh *LDA*. Maka dari itu, pada tugas akhir ini menggunakan *Gaussian Latent Dirichlet Allocation* (*GLDA*) pada *author-topic* model untuk menyelesaikan permasalahan yang ada pada *LDA* dengan mengubah jenis kata menjadi vektor kata. Metode *GLDA* ini digunakan untuk pembanding dari

penelitian sebelumnya. Pembuatan *author-topic* model terbaik dari kedua metode dilakukan dengan menghitung nilai *perplexity* dan *topic coherence* untuk setiap metode. Hasil yang didapat menggambarkan bahwa penggunaan *GLDA* dapat mendapatkan hasil modelling yang lebih stabil disetiap iterasi. Namun, hasil yang didapatkan dari kemampuan kedua model dalam memberikan prediksi terhadap *author* dari dokumen baru memiliki persamaan. Pada kedua model terlihat bahwa persentase prediksinya tidak berbeda jauh. Hal tersebut disebabkan karena penggunaan data dengan jenis kata yang terlali sedikit dan pembersihan data yang kurang baik. Sehingga membuat distribusi kata pada setiap topiknya terlihat tidak sesuai dengan topik dari dokumen tersebut.

Kata Kunci: ACL, Author-Topic Model, Gaussian LDA, LDA, OOV

TOPIC ANALYSIS AND AUTHOR IN THE ENGLISH JOURNAL USING THE AUTHOR-TOPIC MODELS AND GAUSSIAN LATENT DIRICHLET ALLOCATION (GLDA)

Student Name	**: Muchammad Andhika Supriyanto**
NRP	**: 05211540000085**
Department	**: Information Systems FTIK-ITS**
Supervisors	**: Renny Pradina K., S.T, M.T., SCJP**

ABSTRACT

The use of Author-Topic Model has been widely used in the field of machine learning, especially natural language processing. This method was first used to find out the research field of an author by looking at the topic of the document he had written. Although this model is widely used, its application still has many disadvantages. The most prominent thing is the inability of Latent Dirichlet Allocation (LDA) as the basis of the Author-Topic Model to capture words that have never been seen in training data when conducting experiments from a test document in determining word distribution of a topic or commonly called out of vocabulary (OOV). OOV makes the topic obtained from a test document not as desired. This can happen because the important words contained in the test document cannot be captured by the LDA. Therefore, in this final assignment use the Dirichlet Allocation Latent Gaussian (GLDA) on the author-topic model to solve the problems that exist on the LDA by changing the type of word into a word vector. This GLDA method is used for comparison from previous studies. Making the best author-topic model of the two methods is done by calculating the perplexity value and topic

coherence for each method. The results obtained illustrate that the use of GLDA can get more stable modeling results in each iteration. However, the results obtained from the ability of the two models to provide predictions to the author of the new document have similarities. In both models it can be seen that the percentage of predictions is not very different. This is due to too little use of data types and poor data cleaning. so that the distribution of words on each topic does not appear to fit the topic of the document.

Keywords: ***ACL, Author-Topic Models, Gaussian LDA, LDA, OOV***

KATA PENGANTAR

Segala puji dan syukur saya ucapkan kepada Allah SWT yang telah melimpahkan banyak kemudahan dan petunjuk sehingga penulis dapat menyelesaikan Tugas Akhir dengan judul "ANALISIS TOPIK DAN AUTHOR PADA JURNAL BAHASA INGGRIS DENGAN MENGGUNAKAN METODE AUTHOR-TOPIC MODELS DAN GAUSSIAN LATENT DIRICHLET ALLOCATION (GLDA)". Semoga Tugas Akhir tersebut memberikan manfaat bagi bidang keilmuan Sistem Informasi, bermanfaat bagi ITS, pemerintah daerah, dan Indonesia.

Dalam pengerjaan Tugas Akhir ini penulis mendapat banyak sekali bantuan dalam berbagai hal. Penulis ingin menyampaikan ucapan terima kasih kepada:

1. Bapak Gatot Supriyanto, Ibu Nadhif Ulfiah, dan Syafa Rossita Firdadhila Supriyanto selaku keluarga yang selalu memberikan dukungan dan motivasi untuk menyelesaikan tugas akhir ini.
2. Ibu Renny Pradina K., S.T, M.T., SCJP selaku dosen pembimbing yang selalu memberikan masukan, kritik, ilmu, dan motivasi dalam pengerjaan tugas akhir.
3. Bapak Radityo Prasetianto W., S.Kom, M.Kom selaku dosen wali yang selalu memberikan pengarahan dan bimbingan.
4. Ibu Mahendrawati ER, ST, M.Sc, Ph.D selaku Kepala Departemen Sistem Informasi yang memberikan bantuan dan dukungan akademik.
5. Bapak Faizal Johan Atletiko, S.Kom, M.T, dan Bapak Radityo Prasetianto Wibowo, S.Kom, M.Kom selaku dosen penguji yang memberikan kritik dan saran terhadap tugas akhir.

6. Bapak Nisfu Asrul Sani, S.Kom, M.Sc selaku kepala program studi sistem informasi yang telah memberikan bantuan atas permasalahan saya.
7. Alumni Sistem informasi yang sudah berkenan memberikan bentuan atas permasalahan saya
8. Seluruh Dosen Departemen Sistem Informasi ITS yang telah memberikan ilmu, pengalaman, dan bantuan yang sangat berarti bagi penulis.
9. Keluarga VSNMC dan LMB ITS yang memberikan kenangan berharga dan pengalaman tidak terlupakan selama menjalani kehidupan kampus.
10. Kotak, Benny Sapi, Faiq yang telah berkenan meminjamkan laptop beserta komputer untuk mengerjakan tugas akhir ini.
11. Yayan, Irul, Edwin dan Agung yang telah memberikan kebahagiaan dengan mengajak saya bermain kartu setiap malam.
12. Aji, Jody dan Kotak sebagai teman kos dan mencari makan selama saya kuliah.
13. Kacang, Azzam, Salim, Evia, Magrid, Weny, dan Andira selaku teman bimbingan yang selalu memberi semangat.
14. Teman – teman Lannister 2015 yang selalu mengingatkan penulis untuk menyelesaikan tugas akhir.
15. Serta semua pihak yang tidak dapat disebutkan satu per satu oleh penulis yang juga membantu dalam penyelesaian tugas akhir.

Tugas akhir ini membutuhkan banyak sekali masukan dan saran yang membangun dalam rangka penyempurnaan isi dan materi. Namun penulis memiliki harapan bahwa tugas akhir ini bermanfaat bagi pembaca dan perkembangan bidang keilmuan Sistem Informasi

DAFTAR ISI

Halaman ini sengaja dikosongkan

DAFTAR GAMBAR

Halaman ini sengaja dikosongkan

DAFTAR TABEL

Halaman ini sengaja dikosongkan

DAFTAR KODE

DAFTAR DIAGRAM

Halaman ini sengaja dikosongkan

BAB I
PENDAHULUAN

Pada bab ini akan dijelaskan tentang latar belakang, rumusan masalah, Batasan masalah, tujuan dan relevansi dari penelitian yang dilakukan.

1.1 Latar Belakang

Topik penelitian yang dilakukan oleh peneliti semakin beragam pada era ini. Terkadang suatu topik penelitian dapat menginisiasi terbentuknya cabang topik penelitian yang lain dengan tujuan untuk memperbaiki atau menciptakan hal baru dari penelitian sebelumnya. Hal tersebut dapat dilihat pada penelitian terbaru yang melakukan sitasi pada penelitian sebelumnya.

Seperti halnya Association for Computational Linguistics (*ACL*) yang dilakukan setiap tahun. *ACL* merupakan konferensi yang mengkaji perkembangan ilmu dalam bidang natural language (*NL*). Pada konferensi ini, tidak semua penelitian dapat dibahas dalamya. Hanya penelitian yang telah dinyatakan lolos tahap review. Tahap review ini dilakukan oleh peniliti yang memiliki bidang minat yang sesuai dengan penelitian.

Author-topic model (*ATM*) adalah sebuah metode untuk mengetahui topik-topik yang ada pada dokumen yang ditulis oleh peneliti. *ATM* sendiri merupakan gabungan dari 2 metode sebelumnya, yaitu topic model dan author model yang keduanya berbasis menggunakan *Latent Dirichlet Allocation* (*LDA*) [1]. Masalah utama yang ada pada *LDA* terdapat dalam menentukan distribusi kata pada setiap topiknya. LDA belum mampu menangkap kata-kata baru yang mungkin ada pada dokumen uji atau biasa disebut *out of vocabulary* (*OOV*) [2]. *OOV* membuat topik yang didapat dari suatu dokumen uji tidak sesuai dengan yang diinginkan. Hal tersebut dapat terjadi karena kata-kata penting yang terdapat pada dokumen uji belum terdapat pada model sehingga membuat kata tersebut tidak

dikenali oleh LDA walaupun kata-kata tersebut merupakan poin penting pada dokumen tersebut.

Sehingga diusulkan untuk menggunakan word embeddings atau mengubah jenis kata menjadi vektor kata agar dalam penentuan distribusi kata dalam topik pada dokumen baru akan dapat menangkap kata-kata baru tersebut [2]. Eksplorasi yang akan dilakukan pada penelitian ini yaitu menggunakan word embedding berbasis skip-gram untuk melakukan *ATM*. Salah satu metode yang menggunakan word embedding yaitu *Gaussian Latent Dirichlet Allocation* (*GLDA*). Dimana *GLDA* akan mengubah kata menjadi vektor kata dalam embedding space. Tujuannya agar model dapat menangkap kata-kata yang belum pernah terlihat sebelumnya dengan mencari vektor kata dari kata baru terdekat yang ada dalam model pelatihan.

1.2 Perumusan Masalah

Berdasarkan uraian latar belakang yang telah disampaikan, berikut merupakan perumusan masalah pada tugas akhir ini,

- Bagaimana melakukan *Author-Topic* model dengan menggunakan *GLDA* untuk mengatasi *OOV* pada *LDA* dalam menganalisa persebaran kata untuk setiap topik dalam dokumen?
- Bagaimana pengaruh penggunaan *GLDA* pada metode *Author-Topic* model terhadap hasil yang diperoleh?
- Bagaimana perbandingan hasil menggunakan data yang lebih banyak dan spesifik dengan data yang lebih sedikit dan general?

1.3 Batasan Masalah

Berikut ini Batasan masalah tugas akhir ini, berdasarkan latar belakang dan perumusan masalah yang telah dijelaskan,

- Data yang digunakan untuk pengerjaan tugas akhir ini adalah data teks yang tersimpan pada *ACL Anthology*

- Data yang digunakan hanya dari tahun 2014 hingga tahun 2018
- Data yang digunakan untuk pemodelan adalah data yang diubah menjadi format *txt* dengan mengambil teks mulai dari abstrak hingga kesimpulan.

1.4 Tujuan

Berdasarkan perumusan dan Batasan masalah yang telah diuraikan sebelumnya, maka tujuan yang ingin dicapai adalah,

- Melakukan eksperimen *ATM* menggunakan *GLDA* untuk mengatasi *OOV*
- Melakukan analisis dari eksperimen untuk mengetahui hasil perbandingan dari penggunaan *GLDA* dan *LDA* pada *ATM*

1.5 Manfaat

Berikut merupakan manfaat yang diharapkan dari penelitian yang dilakukan:

- Memahami penerapan dan manfaat GLDA pada metode Author-Topic model
- Melakukan eksperimen untuk mengetahui kinerja pemodelan menggunakan GLDA.

1.6 Relevansi

Tugas akhir ini berhubungan dengan mata kuliah Matematika Diskrit, Sistem Cerdas, Bahasa Pemrograman. Serta berhubungan dengan mata kuliah pilihan yang ada pada laboratorium Akuisisi Data dan Diseminasi Informasi (ADDI), yaitu Pengolahan Bahasa Alami.

Halaman ini sengaja dikosongkan

BAB II
TINJAUAN PUSTAKA

Pada bab ini berisi penelitian sebelumnya yang dijadikan acuan dalam pengerjaan tugas akhir dan juga berisi dasar teori untuk menunjang penelitian pada tugas akhir.

2.1 Penelitian Sebelumnya

2.1.1 Improving Distributed Word Representation and Topic Model by Word-Topic Mixture Model

Tabel 2.1 Literatur 1

Nama Peneliti	Xianghua Fu, Ting Wang, Jing Li, Chong Yu, Wangwang Liu
Metodologi	• *Topic model* • Distribusi kata • *Word embedding* • *Word topic mixture model*
Relevansi Penelitian	Penelitian berfokus pada analisis topic model dalam menentukan distribusi kata pada topik. Analisis dilakukan dengan perbandingan metode topic model menggunakan *LDA* dengan menggunakan word embedding. Didapatkan hasil bahwa menggunakan *word embedding* akan mendapatkan hasil yang lebih baik. Selain itu, penelitian ini juga melakukan perbandingan metode topic model menggunakan *word embedding* dengan data yang jumhlahnya berbeda. Didapatkan hasil bahwa menggunakan data yang lebih

	banyak akan mendapatkan hasil yang lebih baik.

2.1.2 Gaussian LDA for Topic Model with Word Embeddings

Tabel 2.2 Literatur 2

Nama Peneliti	R. Das, M. Zaheer, C. Dyer
Metodologi	• *LDA* • *Topic model* • *Word embedding* • *Gassian LDA*
Relevansi Penelitian	Penelitian ini berfokus pada implementasi metode terbaru GLDA dari perkembangan metode sebelumnya yaitu LDA. Hal tersebut dilakukan dengan tujuan untuk menangani permasalahan yang terdapat pada LDA pada saat penentuan distribusi kata pada topik yang ditemukan pada suatu dokumen, yaitu tidak bisa dalam menangani Out of Vocabulary (OOV). Sehingga diusulkan metode GLDA dengan cara mengubah kata menjadi vector sehingga kata-kata yang tidak terlihat sebelumnya dapat dicapai dengan mencari vector terdekat dari kata tersebut.

2.1.3 The Author-Topic Model for Authors and Documents

Tabel 2.3 Literatur 3

Nama Peneliti	M. Rosen-Zvi, T. Griffiths, M. Steyvers, P. Smyth
Metodologi	• *LDA* • *Author-Topic Model*
Relevansi Penelitian	Penelitian ini mengenalkan sebuah metode Author-Topic Model yang digunakan untuk melihat topik penelitian yang dilakukan peneliti pada sebuah dokumen. Hal tersebut relevan dan menjadi dasar dari penelitian ini. Melihat kekurangan yang ada pada metode ini, sehingga pada penelitian pada tugas akhir ini akan melakukan sebuah metode yang akan mengatasi permasalahn yang ada pada Author-Topic Model

2.2 Dasar Teori

2.2.1 *Natural Language Processing (NLP)*

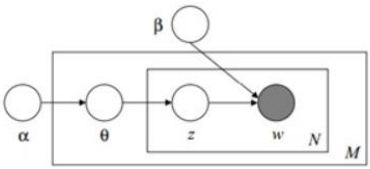
Natural Language Processing (*NLP*) adalah salah satu bidang ilmu komputer, kecerdasan buatan, dan bahasa (linguistik) yang berkaitan dengan interaksi antara komputer dan bahasa alami manusia, seperti bahasa Indonesia atau bahasa Inggris. Tujuan utama dari studi *NLP* adalah membuat mesin yang mampu mengerti dan memahami makna bahasa manusia lalu memberikan respon yang sesuai.

Sejarah *NLP* dimulai pada tahun 1950-an, meskipun telah ada penilitian *NLP* pada tahun-tahun sebelumnya. Pada tahun 1950, Alan Turing mempublikasikan artikel terkenalnya yang berjudul "*Computing Machinery and Intelligence*" yang di

dalamnya Alan Turing mengusulkan tes yang sekarang disebut dengan Turing Test. Tes Turing adalah sebuah tes yang mengukur kemampuan mesin untuk menunjukan perilaku cerdas. Dalam ilustrasi contoh aslinya, seorang juri manusia akan terlibat dalam percakapan dengan manusia dan mesin yang akan dites. Semua peserta dipisahkan satu sama lain. Jika juri tidak bisa membedakan antara manusia dan mesin, maka mesin tersebut dikatakan lulus tes.

2.2.2 *Latent Dirichlet Allocation (LDA)*

Latent Dirichlet Allocation (LDA) merupakan model Bayesian hirarki tiga tingkat, dimana setiap item koleksi dimodelkan sebagai campuran terbatas atas serangkaian set topik. Setiap topik dimodelkan sebagai campuran tak terbatas melalui set yang mendasari probabilitas topik. Dalam konteks pembuatan model teks, probabilitas topik memberikan representasi eksplisit dari sebuah dokumen. Ide dasar dari LDA yaitu dokumen terdairi dari beberapa topik [3].



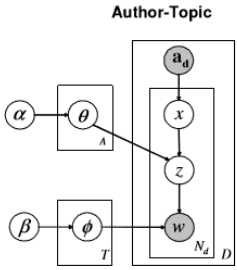
Gambar 2.1 Arsitektur LDA [3]

Terdapat tiga tingkatan dalam *LDA* pada Gambar 2.1, yaitu tingkatan korpus, dokumen, dan kata. Parameter α dan β termasuk dalam tingkatan korpus, diasumsikan sebagai sampel sekali dalam proses menghasilkan korpus. Variabel θ_d termasuk dalam tingkatan dokumen. Variabel z_dn dan w_dn termasuk dalam tingkatan kata yang akan diambil sampel satu kali per dokumen.

2.2.3 Author Topic Model

Author-topic modeling (*ATM*) adalah sebuah metode untuk mengetahui topik-topik yang ada pada dokumen yang ditulis oleh peneliti. *ATM* sendiri merupakan gabungan dari 2 metode

sebelumnya, yaitu topic modelling dan author modelling yang keduanya berbasis menggunakan *Latent Dirichlet Allocation* (*LDA*) [1]. Ide dasar dalam melakukan *ATM* adalah melakukan *Author Modelling* untuk menentukan *Author*, lalu dikombinasikan dengan topic modelling untuk menentukan topik dan distribusi kata.

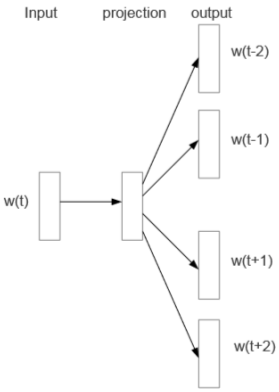


Gambar 2.2 Arsitektur *ATM* [1]

Pada Gambar 2.2 dapat dilihat bahwa sekelompok *author*(ad) memutuskan untuk menulis dokumen(D). Untuk setiap kata dalam dokumen, *author* dipilih secara acak. Kemudian, topik dipilih dari distribusi topik-topik yang berkaitan dengan author, dan kata-kata dihasilkan dari topik yang dipilih [1].

2.2.4 Word Embedding

Word Embedding adalah salah satu metode untuk merepresentasikan kata dengan kelebihan strukturnya yang padat dan dipercaya dapat memberikan makna semantik. Word Embedding mampu menangkap konteks kata dalam dokumen, kesamaan semantik dan sintaksis, hubungannya dengan kata lain, dan lain lain [4]. Ide dasarnya adalah merepresentasi vektor dari kata tertentu. Word2Vec adalah salah satu teknik yang paling populer untuk mempelajari word embedding dengan menggunakan neural network. Terdapat 2 metode, yaitu Skip-gram dan Common Bag of Word(CBOW). Pada penelitian ini menggunakan metode Skip gram dikarenakan lebih baik dalam mengatasi permasalahan kata-kata yang jarang muncul dibandingkan dengan CBOW [5].



Gambar 2.3 Arsitektur *Skrip-gram* [5]

Tujuan pelatihan dari model *Skip-gram* yang terlihat pada Gambar 2.3 adalah untuk menemukan representasi kata yang berguna untuk memprediksi kata-kata di sekitarnya dalam kalimat atau dokumen [6]. Jika sebuah dokumen dilambangkan dengan D, kata dan indeks kata dilambangkan dengan w_i. Maka sebuah dokumen dapat dimodelkan D={w_1, w_2, w_3, . . . , w_n}. Proses pelatihan pada skip gram adalah memberikan nilai window t untuk setiap w_i yang nantinya akan digunakan sebagai masukan oleh neural network untuk menebak sekumpulan kata dari w_(t-i), … , w_(t+i).

2.2.5 *Gaussian Latent Dirichlet Allocation (GLDA)*

GLDA merupakan variasi dari *LDA* yang beroperasi pada embeddings space secara terus menerus dari kata-kata, dan bukan tipe kata yang dilakukan *LDA*. Pendekatan ini menggantikan jenis kata samar atau belum terlihat sebelumnya saat dimodelkan di *LDA* dengan embedding space secara terus menerus dari kata-kata ini, yang dihasilkan sebagai gambar dari *Gaussian multivarian* [7]. Model ini dapat mengeksplitasi kata-kata yang mirip secara semantic di embedding space dan dapat menetapkan probabiltas topik yang tinggi untuk sebuah kata yang mirip dengan kata yang ada, bahkan jika belum pernah terlihat sebelumnya [2].

2.2.6 Validasi Topik Berdasarkan Nilai Perplexity dan Topic Coherence

Perplexity berarti ketidakmampuan untuk memahami sesuatu yang rumit. Dalam NLP digunakan untuk mengukur tingkat ketidakpastian yang dimiliki model dalam memprediksi (probabilitas) beberapa teks. Suatu model dikatakan baik jika memiliki nilai perplexity yang kecil [8]. Artinya adalah ketika nilai perplexity kecil, maka model tersebut akan memiliki probabilitas yang tinggi. Sehingga dapat membuat topik-topik yang ditemukan berdasarkan distribusi kata akan semakin berbeda.

Dalam mendeskripsikan dokumen dengan baik dengan mencapai perplexity rendah, diinginkan untuk memiliki topik yang dapat ditafsirkan oleh manusia. Namun secara mengejutkan perplexity dan penilaian manusia tidak berkorelasi, dan bahkan kadang-kadang sedikit berkorelasi [9]. Topic coherence merupakan ukuran yang digunakan untuk mengevaluasi sebuah topik yang dihasilkan oleh model agar dapat dipahami oleh manusia. Model yang baik akan menghasilkan nilai topic coherence yang tinggi [10].

2.2.7 Perhitungan Nilai Similarity Matrix

Hellinger distance merupakan suatu metode untuk menghitung nilai similarity berbasis probabilistik. Ide dasarnya menghitung tingkat kemiripan dua objek dengan merepresentasikan dua set objek yang diperbandingkan tersebut dalam bentuk probability [11]. Diberikan dua distribusi probabilitas diskrit P = (p_1, ... , p_k) dan Q = (q_1, ... , q_k), Hellinger distance H(P,Q) antara distribusi didefinisikan sesuai dengan persamaan (1):

$$H(P,Q) = \frac{1}{\sqrt{2}}\sqrt{\sum_{i=1}^{k}(\sqrt{p_i} - \sqrt{q_i})^2} \quad (1)$$

Selain menggunakan metode Hellinger distance untuk menhitung nilai similarity matrix. Pada penelitian ini juga menggunakan beberapa metode sebagai pembanding metode Heliinger distance. Metode yang dipakai adalah cosine similarity dan metode similarity dari dua nilai integer.

Cosine similarity merupakan pendekatan yang digunakan untuk menentukan kemiripan dari dokumen-dokumen terlepas dari besar kecilnya dokumen tersebut. Secara matematis, pendekatan ini mengukur sudut kosinus antara dua vector yang diproyeksikan dalam ruang multi-dimensi. Dalam konteks ini, dua vektor ini merupakan array yang berisi jumlah kata dari dua dokumen, seperti terlihat pada persamaan(2).

$$Cos\theta = \frac{\vec{a}.\vec{b}}{||\vec{a}||||\vec{b}||} = \frac{\sum_i^n a_i b_i}{\sqrt{\sum_i^n a_i^2}\sqrt{\sum_i^n b_i^2}} \quad (2)$$

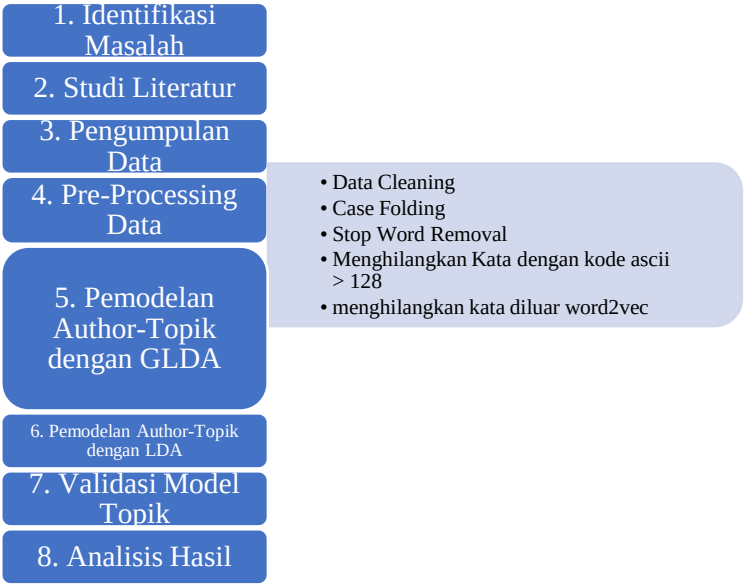
Ketika diplot pada ruang multi-dimensi, di mana setiap dimensi sesuai dengan kata dalam dokumen, cosine similarity menangkap orientasi(sudut) dokumen, bukan besarnya. Cosine similarity menguntungkan karena walaupun kedua dokumen yang memiliki topik sama tetap terpisah jauh oleh jarak Euclidean karena ukurannya, kedua dokumen ini masih dapat memiliki kesamaan atau mendapatkan suduh yang kecil. Dimana, semakin kecil sudut dari dua dokumen maka semakin tinggi kesamaannya.

Sedangkan untuk mengukur tingkat kesamaan antara dua dokumen, penulis juga mengusulkan pendekatan yang dilakukan secara manual dengan menghitung kesamaan dari dua integer. Dimana integer yang dimaksudkan adalah nilai yang dimiliki oleh masing masing dokumen. Pendekatan ini dirasa efektif, dikarenakan hanya menghitung kesamaan dari dua integer saja tanpa menambahkan parameter-parameter lainnya.

BAB III
METODOLOGI

Pada bab ini dijelaskan metodologi yang digunakan dalam penelitian beserta penjelasan setiap langkah tersebut dan juga jadwal pengerjaan tugas akhir.

3.1 Metodologi



Gambar 3.1 Tahapan Pelaksanaan

Berikut merupakan langkah – langkah penelitian:

a. Identifikasi Masalah

Pada tahap ini dilakukan identifikasi masalah dengan cara mengobservasi permasalahan pada journal conference dan Author-topik model. Hasil observasi menunjukkan bahwa tidak seluruh journal dapat lolos atau dapat mengikuti conference. Hanya journal yang telah lolos uji review dengan reviewer yang sesuai dengan topik dari journal

tersebut. Penunjukan reviewer yang sesuai dengan topik tidak luput dari metode Author-Topic model dalam mengetahui topik penelitian dari dokumen yang dibuat oleh peneliti. Dari observasi yang dilakukan dengan menganalisa penelitian sebelumnya, terdapat kekurangan yang dimiliki oleh Author-Topic model. Metode ini masih dirasa kurang dalam menangani *Out of Vocabulary (OOV)* dalam menentukan distribusi kata pada setiap topik. Berdasarkan kondisi tersebut, dirasa perlu melakukan penggunaan metode yang dirasa dapat mengatasi masalah tersebut.

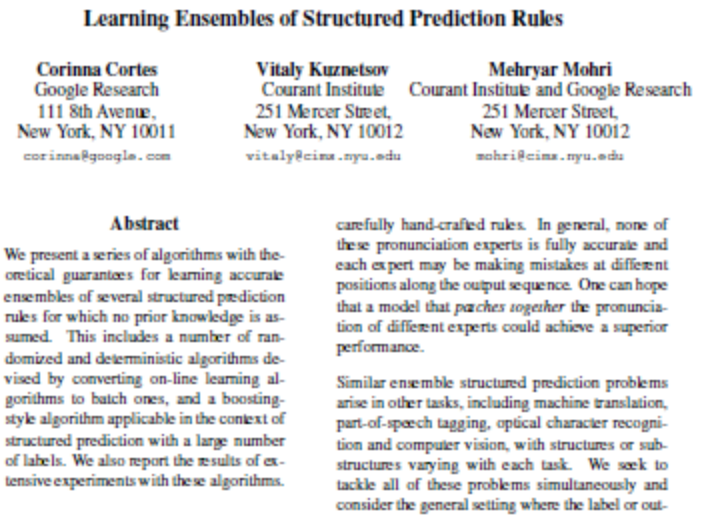
b. Studi Literatur

Pada tahap ini dilakukan proses pemahaman terhadap konsep, metode, dan permasalahan yang ada pada penelitian sebelumnya pada sebuah metode yang dilakukan. Tahap ini dilakukan untuk menggali informasi terkait dengan permasalahan dan solusi yang dibutuhkan untuk penelitian ini. Literatur yang digunakan untuk mengatasi permasalahan penentuan topik pada sebuah dokumen mengacu pada penelitian M. Rosen-Zvi dkk dengan judul "The Author-Topic Model for Document and Author". Pembahasan mengenai solusi untuk permasalahan Author-Topic model mengacu pada penelitian R. Das dkk yang berjudul "Gussian LDA for Topic Model". Pembahasan mengenai jumlah data terbaik yang digunakan dalam penentuan distribusi kata pada topik mengacu pada penelitian Xianaghua Fu dkk yang berjudul "Improving Distributed Word Representation and Topic Model by Word-Topic Mixture Model".

c. Pengumpulan Data

Tahap ini dilakukan proses pengumpulan, pemahaman, dan pemilihan data yang digunakan untuk dianalisis. Data yang akan dianalisis merupakan journal online berbahasa inggris. Pengumpulan data dilakukan dengan mengunduh journal dari archive *ACL Anthology* dengan format *pdf*. Data yang dikumpulkan hanya data dari tahun 2014 hingga

tahun 2018. Contoh data yang diunduh dari *ACL Anthology* dengan format *.pdf* dapat dilihat pada Gambar 3.2



Learning Ensembles of Structured Prediction Rules

Corinna Cortes
Google Research
111 8th Avenue,
New York, NY 10011
corinna@google.com

Vitaly Kuznetsov
Courant Institute
251 Mercer Street,
New York, NY 10012
vitaly@cims.nyu.edu

Mehryar Mohri
Courant Institute and Google Research
251 Mercer Street,
New York, NY 10012
mohri@cims.nyu.edu

Abstract

We present a series of algorithms with theoretical guarantees for learning accurate ensembles of several structured prediction rules for which no prior knowledge is assumed. This includes a number of randomized and deterministic algorithms devised by converting on-line learning algorithms to batch ones, and a boosting-style algorithm applicable in the context of structured prediction with a large number of labels. We also report the results of extensive experiments with these algorithms.

carefully hand-crafted rules. In general, none of these pronunciation experts is fully accurate and each expert may be making mistakes at different positions along the output sequence. One can hope that a model that *patches together* the pronunciation of different experts could achieve a superior performance.

Similar ensemble structured prediction problems arise in other tasks, including machine translation, part-of-speech tagging, optical character recognition and computer vision, with structures or substructures varying with each task. We seek to tackle all of these problems simultaneously and consider the general setting where the label or out-

Gambar 3.2 Contoh Data dengan Format PDF

Data yang berhasil diunduh tersebut diubah kedalam bentuk *.txt.* pengubahan format data dilakukan dengan menggunakan *converter online*. Setelah itu dilakukan juga pengubahan secara manual untuk mengambil teks dari abstract hingga kesimpulan saja, Contoh data yang berhasil dilakukan pengubahan format dapat dilihat pada gambar

```
Learning Ensembles of Structured Prediction Rules

Abstract

carefully hand-crafted rules. In general, none of
these pronunciation experts is fully accurate and
each expert may be making mistakes at different
positions along the output sequence. One can hope
that a model that patches together the pronunciation of different experts could achieve
a superior
performance.

We present a series of algorithms with theoretical guarantees for learning accurate
ensembles of several structured prediction
rules for which no prior knowledge is assumed. This includes a number of randomized and
deterministic algorithms devised by converting on-line learning algorithms to batch
ones, and a boostingstyle algorithm applicable in the context of
structured prediction with a large number
of labels. We also report the results of extensive experiments with these algorithms.
```

Gambar 3.3 Contoh Data dengan Format TXT

d. Pre-Processing Data

Pada tahap ini dilakukan langkah-langkah pembersihan data, sehingga data yang digunakan untuk topic modeling sesuai dan siap diolah. langkah-langkah dari tahap ini dalah sebagai berikut,

i. *Data Cleaning*

Pada tahap data cleaning, dilakukan pembersihan data dari angka dan simbol yang tidak digunakan. Seperti halnya sitasi yang menggunakan satu digit angka. Tujuan dari data cleaning untuk mengurangi tingginya frekuensi dari kemunculan karakter agar tidak berpengaruh pada distribusi kata pada setiap topik.

ii. *Case Folding*

Teks sering kali memiliki beragam kapitalisasi yang mencerminkan awal kalimat, penekanan kata benda yang tepat. Case folding dilakukan untuk merubah teks menjadi huruf kecil. Tujuannya untuk menghilangkan perbedaan pada kata yang sama namun memiliki kode binary yang berbeda pada mesin.

iii. *Stop Word Removal*

Pada tahap ini dilakukan penghilangan kata yang sering muncul dalam teks namun dianggap tidak memiliki makna. Biasanya kata yang termasuk adalah kata awalan, kata depan, dan kata penghubung. Tujuannya untuk mengurangi tingginya frekuensi kata yang akan berdampak pada penentuan distribusi kata pada topik.

iv. Penghapusan Kata dengan Kode *ASCII* > 128

Pada tahap ini dilakukan pembersihan data dari kata dengan kode *ascii* lebih dari 128. Tujuan dari penghapusan kata ini untuk menghilangkan huruf dengan bahasa yang tidak sesuai dengan alphabet dan tidak hilang pada saat proses *data cleaning.*

v. Penghapusan Kata yang Tidak Ada Dalam *word2vec* model

Tahapan ini dilakukan untuk menghilangkan kata-kata yang tidak ada dalam *word2vec model*. Proses ini dilakukan untuk seluruh data yang akan digunakan pada pemodelan *author-topic* menggunakan *LDA* maupun Gaussian *LDA*. Penghapusan kata dilakukan untuk mengurangi kata-kata kotor yang terlewat pada proses sebelumnya.

e. Pemodelan *Author-Topic* dengan *Gaussian Latent Dirichlet Allocation (GLDA)*

Pada tahap ini bertujuan untuk menghasilkan model dengan menggunakan *Gaussian Latent Dirichlet Allocation* (*GLDA*) pada metode *Author-Topic Models*. Dimana *GLDA* digunakan untuk penentuan distribusi kata pada *topic* models. Penentuan model terbaik dilakukan dengan cara menghitung nilai *perplexity* dan *topic coherence.* Hal tersebut dilakukan untuk dibandingkan dengan penelitian sebelumnya yang menggunakan metode *Author-Topic Models*.

f. Pemodelan *Author-Topic* dengan *Latent Dirichlet Allocation (LDA)*

Tahap ini dilakukan untuk menghasilkan *author-topic* model dengan menggunakan Latent Dirichlet Allocation (LDA) pada metode Author-Topic Models. Pada tahap ini juga dilakukan perhitungan nilai *perplexity* dan *topic coherence* dari setiap model untuk mencari model yang terbaik. Pemodelan ini dilakukan sebagai pembanding dari pemodelan sebelumnya

g. Validasi Model *Author-Topic*

Validasi dilakukan untuk mengetahui kemampuan model dalam memprediksi topik penelitian seorang peneliti berdasarkan distribusi kata dan peneliti. Pada tahap ini dilakukan dengan mengukur nilai perplexity dan topic coherence setiap model dengan nilai yang terbaik. Dimana pada nilai perplexity dicari nilai terendah, sedangkan pada topic coherence dicari nilai tertinggi. Kemudian dilakukan perhitungan nilai similarity topik penelitian dengan peneliti pada suatu dokumen yang ada berdasarkan model yang telah terbentuk dengan menggunakan Hellinger Distance.

h. Analisis Hasil

Pada tahap ini dilakukan analisa hasil dari pengujian model terbaik yang didapatkan baik menggunakan *Gaussian LDA* maupun *LDA*. Analisa hasil tersebut didapat dari pengujian nilai kemiripan antara topik pengelitian dengan peneliti yang menulis dokumen tersebut berdasarkan model yang telah terbentuk dengan menggunakan metode *Hellinger Distance, Cosine Similarity, Topic Similarity*. Dari hasil yang didapat dari masing-masing model, nantinya akan dibandingkan keduanya untuk mendapatkan kesimpulan metode yang terbaik dalam melakukan *Author-Topic Model*.

BAB IV
PERANCANGAN

Bab ini menjelaskan perancangan dari penelitian tugas akhir. Perancangan ini digunakan sebagai pedoman dalam melakukan penelitian tugas akhir yang dijelaskan sebagai berikut.

4.1 Pengambilan Data

Untuk dapat mengetahui informasi topik yang terdapat pada jurnal penelitian *Association for Computational Linguistics (ACL)*, dibutuhkan data dengan tipe teks yang berasal dari website repositori jurnal penelitian *ACL (ACL Anthology).* Pengambilan data dilakukan dengan cara mengunduh data dari *ACL Anthology*. Data yang diunduh merupakan data dengan jarak waktu 2014 hingga tahun 2018. Jumlah data yang terunduh sebanyak 1003 jurnal penelitian berbahasa inggris.

Data berekstensi *.pdf* yang telah diunduh akan diolah dengan hanya mengambil nama penulis (*author)* dan isi dari dokumen tersebut yang hanya diambil dari abstrak hingga kesimpulan. Hal tersebut dikarenakan terdapat banyak kata yang tidak terlalu relevan jika diproses dalam topik model seperti alamat, email, ucapan terimakasih, dan referensi.

4.2 Metodologi Implementasi Penelitian

Metodologi implementasi penelitian merupakan tahapan yang dilakukan untuk mencapai tujuan penelitian yang dilakukan dengan komputasi secara otomatis. Beberapa tahap yang dilakukan pada penelitian ini yakni mempersiapkan data, pra-proses data, pemrosesan data, pemodelan author-topic menggunakan Gaussian LDA, pemodelan author-topic menggunakan LDA, dan validasi model dari kedua pemodelan.

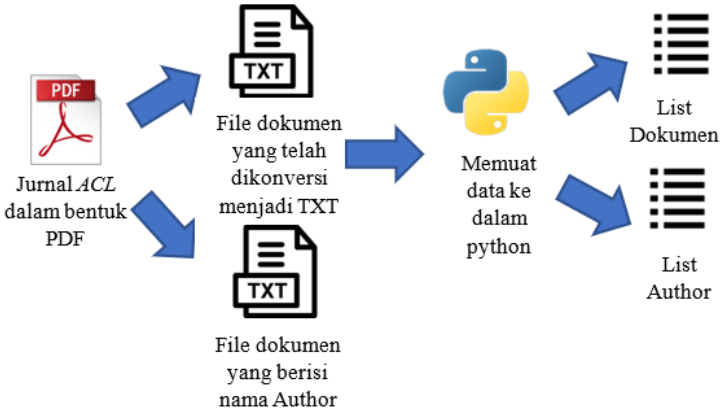
4.2.1 Mempersiapkan Data

Tahap mempersiapkan data merupakan tahapan untuk mengolah data yang ada agar struktur datanya sesuai dengan data yang dibutuhkan untuk melakukan pemrosesan data dan

analisis. Tahap pertama yang dilakukan adalah mengubah format data dari *.pdf* menjadi format *.txt* dengan menggunakan *converter online*. Setelah dokumen selesai dilakukan pengubahan, lalu hasil dokumen tersebut dilakukan pembersihan secara manual untuk mengambil data dari *abstrak* hingga kesimpulan. Alur tahap mengubah format data sesuai dengan Gambar 4.1

Gambar 4.1 Alur Mengubah Format Data

Setelah data berhasil diubah seluruhnya menjadi format *.txt*, selanjutnya dilakukan tahap memuat data ke dalam *python.* Alur tahap memuat data sesuai dengan Gambar 4.2



Gambar 4.2 Alur Mempersiapkan Data

Data mentah dalam format *.pdf* dikonversi kedalam dua *file* dengan format *.txt*. File dokumen yang merupakan isi dari jurnal

dikonversi menggunakan *converter online* sesuai dengan Gambar 4.1. File dokumen yang berisi nama Author dikonversi kedalam format *.txt* secara manual. Kedua hal tersebut dilakukan untuk memudahkan proses pengolahan data menggunakan *python.* Selanjutnya data yang telah dalam format *.txt* kemudian dimuat kedalam variabel dalam bentuk list(dokumen) dan dictionary(*author*) dengan menggunakan *python*.

4.2.1.1 Membuat List Dokumen

Isi dari jurnal *ACL* yang telah dikonversi ke dalam bentuk .txt kemudian dimuat menjadi bentuk variabel list. Nama dari masing-masing dokumen harus sesuai dengan *ID* dokumen yang tercantum pada list *author*.

4.2.1.2 Membuat List Author

Untuk mendapatkan *list author*, dilakukan pengambilan nama author dari jurnal *ACL* yang berformat *.pdf* secara manual ke dalam format *.txt*. Isi dari file *.txt* merupakan nama author dengan judul dokumen yang ditulis oleh author. Hal tersebut dilakukan untuk mengetahui topik dari *author* dalam pemrosesan data nantinya. Contoh dari file list author sesuai dengan Gambar 4.3

```
Bansal, Mohit, 98
Bao, Junwei, 91
Baroni, Marco, 9, 23, 59, 132
Barzilay, Regina, 19, 26, 130
Baumel, Tal, 86
Belanger, David, 56
Beltagy, Islam, 114
```

Gambar 4.3 *File Author*

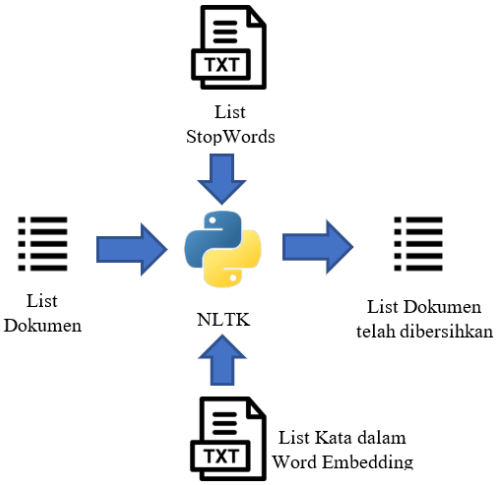
4.2.1.3 Membuat Hubungan antara *Author* dengan Dokumen

Pembuatan *author2doc* dilakukan dengan memetakan tiap dokumen yang ditulis oleh tiap *author* kedalam bentuk dictionary. *Author2doc* ini nantinya digunakan untuk

mengetahui distribusi topik dari tiap author setelah model dan daftar topik terbentuk.

4.2.2 Pra-proses Data

Tahapan pra-proses data merupakan tahapan dasar yang harus dilakukan dalam melakukan pengolahan Bahasa alami. Tujuannya agar data yang digunakan sesuai dengan kebutuhan model. Pada penelitian ini dilakukan tahapan pra-proses data seperti mengubah data menjadi huruf kecil, menghilangkan kata umum(stopwords), menghilangkan kata yang jarang muncul, menghilangkan simbol maupun angka, menghilangkan kata diluar list word2vec model, dan menghilangkan kata dengan kode ascii lebih dari 128. Alur pra-proses data dapat dilihat pada Gambar 4.4



Gambar 4.4 Alur Pra-proses

Langkah pertama yang dilakukan pra-proses pada penelitian ini adalah menghilangkan kata yang tidak ada dalam list *word embedding*. List kata *word embedding* didapatkan dari mengunduh file *word2vec* yang akan dijelaskan secara rinci pada bab pemrosesan vektor. Menghilangkan kata tersebut dilakukan untuk menyamakan teks data yang akan digunakan dalam pemodelan *Author-Topik* menggunakan *Gaussian LDA*

dan *LDA*. Setelah langkah tersebut dilakukan, maka selanjutnya tahapan mengubah data menjadi huruf kecil, menghilangkan teks kata yang jarang muncul, menghilangkan simbol maupun angka, menghilangkan kata dengna kode *ascii* lebih dari 128, dan menghilangkan kata umum. Untuk menghilangkan kata umum dilakukan dengan mendefinisikan *stopwords* terlebih dahulu yang didapatkan dari *library nltk*. Data yang dilakukan pra-proses data hanyalah data list dokumen, baik dokumen yang digunakan untuk training maupun dokumen yang digunakan untuk tahap pengujian.

Setelah seluruh proses pembersihan data dilakukan, maka data yang dianggap sudah bersih dan siap untuk digunakan dalam pemrosesan data

4.2.3 Pemodelan *Author-Topik* Menggunakan *Gaussian LDA*

Pada tahap ini dilakukan pemrosesan data dari pembentukan corpus hingga penggunaan metode Author-Topic Models (ATM) menggunakan Gaussian LDA.

4.2.3.1 Pemrosesan *Corpus*

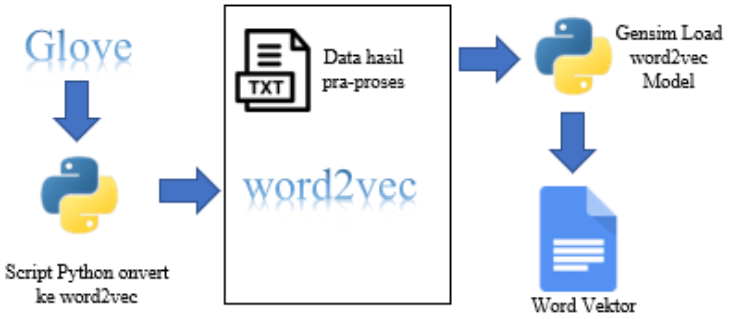
Pada tahap ini dilakukan pemrosesan data menjadi corpus pada data yang telah dilakukan pra-proses sebelumnya. Corpus merupakan dictionary yang berisikan kata dan topik dari kata tersebut yang diberikan secara random.

Selain itu, pada tahap ini juga dilakukan penghapusan kata yang tingkat kemunculannya terlalu sering maupun terlalu jarang pada dokumen. Parameter nilai kata terlalu sering dan kata terlalu jarang dapat diatur secara manual.

4.2.3.2 Pemrosesan Vektor Kata

Pada tahap ini dilakukan pemrosesan kata untuk diubah menjadi bentuk vector. Pengubahan kata menjadi bentuk vektornya dilakukan dengan menggunakan model Glove yang dapat diunduh pada website Stanford. Jumlah kata yang terdapat pada model Glove sebanyak 400.00 kata. Model Glove yang telah diunduh selanjutnya diubah menjadi model word2vec menggunakan library python genism. Hal tersebut dilakukan

untuk mempermudah proses pemodelan topik menggunakan Gaussian Latent Dirichlet Allocation (GLDA). Alur pemrosesan vector kata dapat dilihat pada Gambar 4.5.



Gambar 4.5 Alur Pemrosesan Vektor

4.2.3.3 Proses Pemodelan *Author-Topic* Menggunakan *Gaussian LDA*

Pada tahap ini dilakukan pembuatan model author-topic dengan menggunakan *Gaussian Latent Dirichlet Alloctaion (GLDA)*. Dalam membentuk model, dilakukan beberapa eksperimen pada jumlah topik dan iterasi dimana keduanya adalah parameter masukkan untuk *GLDA*. Model dari beberapa eksperimen tersebut dievaluasi dengan mencari nilai perplexity terbaik, yaitu nilai terkecil. Alur pemodelan *author-topic* dapat dilihat pada Gambar 4.6

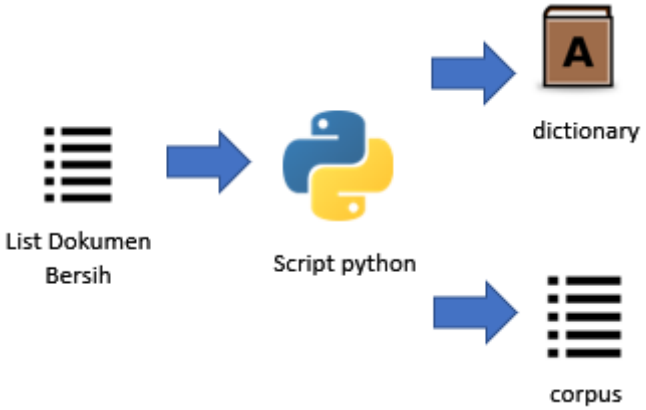
Gambar 4.6 Alur Pemodelan *ATM GLDA*

4.2.4 Pemodelan *Author-Topik* Menggunakan *LDA*

Pada tahap ini dilakukan pemrosesan data dari pembentukan corpus hingga penggunaan metode *Author-Topic Models* (*ATM*) menggunakan *LDA*.

4.2.4.1 Pembuatan *Dictionary* dan *Corpus*

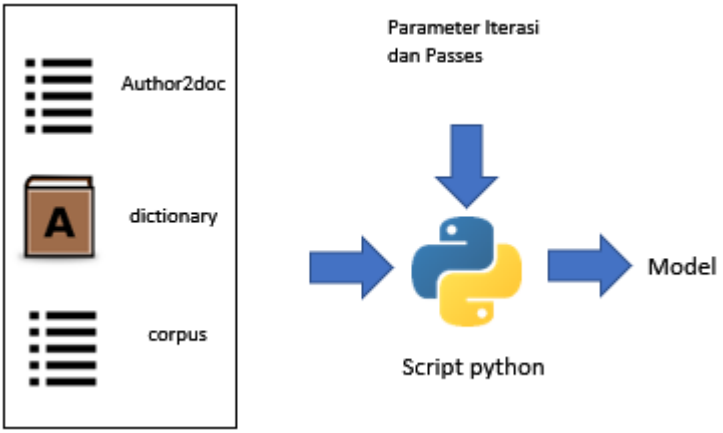
Tahapan ini dilakukan untuk membentuk dictionary dan corpus yang akan digunakan sebagai input dari pemodelan ATM menggunakan LDA. Dictionary dibentuk dari data yang telah dilakukan pre-process sebelumnya. Output dari dictionary adalah kata unik dengan nomor indeksnya. Sedangkan output dari corpus sendiri merupakan format data yang berbentuk bag of word yang merupakan inputan dari pemodelan ATM nantinya. Tahapan ini dilakukan sesuai dengan alur pada Gambar 4.7



Gambar 4.7 Alur Pembuatan Dictionary dan Corpus

4.2.4.2 Proses Pemodelan *Author-Topic* Menggunakan *LDA*

Pemodelan Author-Topic ini menggunakan library atmmodel dari genism. Setelah model terbentuk, nilai perplexity dan topic coherence dari setiap model dijadikan acuan untuk menentukan beberapa model terbaik. Alur Proses tahapan ini sesuai dengan Gambar 4.8



Gambar 4.8 Alur Pemodelan *ATM LDA*

Dilakukan beberapa eksperimen untuk mendapatkan model yang terbaik. Eksperimen pertama adalah menentukan jumlah iterasi(passes) yang didapatkan dari nilai perplexity yang terbaik(kecil). Eksperimen kedua adalah menentukan jumlah topik yang sesuai dengan menghitung rata-rata perplexity dari nilai perplexity terakhir disetiap iterasi yang dilakukan.

4.2.5 Validasi Model

Tahapan ini dilakukan untuk memastikan model yang telah diproses sebelumnya pada tahap pemodelan author-topik pada dokumen yang dihasilkan memiliki nilai probabilitas tertinggi, baik probabilitas persebaran topik maupun probabilitas persebaran kata-kata yang menyusun setiap topik. Berikut hal yang harus diperhatikan dalam tahap validasi model baik menggunakan LDA maupun Gaussian LDA.

1. Jumlah iterasi yang tepat untuk membentuk model berdasarkan nilai perplexity yang dihasilkan
2. Jumlah topik yang tepat untuk membentuk model berdasarkan nilai perplexity yang dihasilkan
3. Jumlah topik yang sesuai berdasarkan nilai topic coherence tiap model

4.2.6 Perancangan Pengujian Model

Pada tahap ini dilakukan pengujian model dari model yang telah dihasilkan pada proses sebelumnya. Hal tersebut bertujuan untuk mengetahui kemampuan masing-masing model dalam melakukan prediksi kemiripan dokumen pengujian dengan author yang menulis dokumen pelatihan.

Terdapat 2 tahapan pada proses perancangan pengujian model ini, yaitu menghitung rata-rata nilai probabilitas kata dalam dokumen untuk setiap topik dan menghitung kemiripan antara dokumen pengujian dengan author berdasarkan matriks probabilitas kata per topik menggunakan 3 metode, yaitu hellinger distance, cosine similarity dan topic similarity.

4.2.6.1 Probabilitas Kata Dalam tiap Topik di Dokumen Uji

Pada tahap ini dilakukan perhitungan rata-rata probabilitas kata dalam dokumen berdasarkan nilai probabilitas kata setiap topik dari hasil topic modelling. Dari perhitungan ini akan menghasilkan nilai matriks yang dapat menggambarkan topik dan probabilitas dari dokumen uji tersebut berdasarkat distribusi kata-katanya pada setiap topik. Dari nilai matriks probabilitas tersebut diurutkan dari nilai yang terbesar. Nilai terbesar tersebut menunjukkan topik yang paling dominan dari dokumen uji tersebut.

4.2.6.2 Perhitungan Kemiripan antara Dokumen Pengujian dengan *Author*

Setelah mengetahui nilai probabilitas dokumen uji untuk setiap topik. Lalu pada tahapan ini akan dilakukan perbandingan dari

hasil kemiripan yang didapatkan dari masing-masing dokumen pengujian dengan Author dengan beberapa metode

4.2.6.2.1 Hellinger Distance

Menghitung kemiripan antara dokumen pengujian dengan author pada metode ini dilakukan dengan menggunakan library genism.matutils.hellinger. Masukkan dari metode ini adalah vektor probabilitas yang dimiliki dokumen dan author pada masing-masing topik. Sedangkan hasil dari metode ini adalah nilai kemiripan antara 0 dan 1. Jika nilainya semakin mendekati nilai 1 maka kedua vektor tersebut semakin mirip. Begitu juga dengan sebaliknya. Penghitungan menggunakan metode ini pada *LDA* dilakukan terlebih dahulu dengan mengubah format vektor probabilitas *bag of word* menjadi array.

4.2.6.2.2 *Cosine Similarity*

Menghitung kemiripan antara dokumen pengujian dengan author pada metode ini dilakukan dengan menggunakan *library genism.matutils.cossim.* masukkan dari metode ini adalah vektor yang berformat *Bag of Word* pada dokumen uji dan author untuk probabilitas masing-masing topik. Penghitunan menggunakan metode ini pada *Gaussian LDA* dilakukan terlebih dahulu mengubah format vektor probabilitas *array list* menjadi *bag of word.*

4.2.6.2.3 *Topic Similarity*

Pada metode ini penulis mengusulkan untuk menghitung tingkat kemiripan antara dua vektor. Dimana dua vektor tersebut didapatkan dari probabilitas terdominan dari dokumen maupun *author* pada topik. Pendekatan ini menggunakan *library difflib.SequenceMatcher*. masukkan dari metode ini adalah vektor probabilitas dari topik terdominan dari dokumen dan *author.*

BAB V
IMPLEMENTASI

Bab ini menjelaskan implementasi dari perancangan yang telah dilakukan sebelumnya. Bagian ini akan menjabarkan hasil eksperimen terhadap data yang menjadi acuan penelitian. Selain itu, akan dijelaskan mengenai pembuatan fitur dalam bentuk kode serta pengujian.

5.1 Perangkat Penelitian

Tahapan ini menjelaskan perangkat yang digunakan selama pengerjaan tugas akhir. Perangkat yang digunakan meliputi perangkat lunak dan perangkat keras. Tabel 5.1 berikut menjelaskan perangkat yang dimaksud:

Tabel 5.1 Spesifikasi Komputer

Windows Based Operating Systems	
Prosesor	Intel® Core™ i5-6500 CPU @3.20Ghz
Memory	16 GB RAM DDR4
Graphic Card	GPU Nvidia GeForce GTX 1060 6GB
Sistem Operasi	*Windows 10 Home*
Arsitektur Sistem	*64-Bit Operating System, x64-based processor*
UNIX Based Operating Systems	
Prosesor	Intel® Core™ i5-8400 CPU
Memory	16 GB RAM DDR4
Graphic Card	GPU Nvidia GeForce GTX 1070 8GB
Sistem Operasi	Linux Ubuntu 16.04
Arsitektur Sistem	64 Bit

Penelitian ini juga dikembangkan dengan menggunakan beberapa software/tools, bahasa pemrograman, dan library yang disajikan dalam Tabel 5.2

Tabel 5.2 Software yang digunakan

Bahasa Pemrograman	*Python* 3.7.0
Software/Tools	• *Sublime Text Editor* • *Jupyter Notebook* • Microsoft Excel 2018
Library	• *Anaconda* • *Gensim* • *NLTK* • *Numpy* • *Sklearn* • *Difflib* • *Re* • *Scipy*

5.2 Mempersiapkan Data

Sebelum dilakukannya pembersihan data untuk pemodelan author-topik. Data yang didapatkan dari mengunduh file dari ACL Anthology dengan berformat pdf diubah terlebih dahulu menjadi format txt. Pengubahan format data tersebut dilakukan dengan menggunakan converter online untuk keseluruhan data yang berjumlah 1023 jurnal penelitian.

File data txt menjadi file data dokumen dengan nama file urutan nomor. Dimana dari data txt tersebut hanya diambil teks dari abstak hingga kesimpulan saja. Sedangkan untuk file data author dibuat secara manual dengan mengambil nama author dari tiap dokumen dan nama dari file dokumen yang ditulis

5.2.1 Memuat Data

Tahapan ini dilakukan pemuatan data yang telah dipersiapkan sebelumnya menggunakan *Jupyter Notebook*. Untuk setiap data

file dokumen yang tersimpan dalam format *txt* akan digabungkan menjadi satu menjadi list dokumen. *List* dokumen diberikan '*id'* sesuai dengan urutan file dokumen tersebut. *List id* dari dokumen akan tersimpan kedalam variabel '*doc_ids'*. Kode 5.1merupakan kode yang digunakan untuk memuat data dokumen menjadi *list id* dokumen.

```
1. data_dir = 'E:/andhika/data/'
2.
3. yrs =['14','15','16','17','18']
4. lokasi = ['test' + '/' + yr for yr in yr
   s]
5. #docs = []
6. doc_ids = []
7.
8. for yr_dir in lokasi:
9.     files = os.listdir(data_dir + yr_dir
   )
10.     for filen in files:
11.         (idx1, idx2) = re.search('[0-
   9]+', filen).span()
12.         doc_ids.append(str(int(filen[idx1:idx2])))
13.
```

Kode 5.1 Memuat Data Dokumen

Setelah memuat list dokumen, selanjutnya memuat data author yang akan dimasukkan kedalam dictionary 'author2doc' yang berisikan key nama dari author dan value list id dokumen yang ditulis oleh author. Kode 5.2 merupakan kode yang digunakan untuk memuat data author menjadi dictionary '*author2doc'*.

```
1. author2doc = dict()
2. i=0
3. for yr in yrs:
4.     filename = data_dir + 'authors/a' +
   yr + '.txt'
5.     for line in smart_open(filename, err
   ors='ignore', encoding='utf-8'):
6.         contents = re.split(',', line)
```

```
7.          author_name = (contents[1] + con
   tents[0]).strip()
8.          author_name = re.sub('\s', '', a
   uthor_name)
9.          ids = [c.strip() for c in conten
   ts[2:]]
10.         if not author2doc.get(author_nam
   e):
11.             author2doc[author_name] = []

12.             i+=1
13.
14.         author2doc[author_name].extend([
   id for id in ids])
```

Kode 5.2 Memuat Data *Author*

Setelah memuat data dokumen dan *author*. Lalu *author* dipetakan dengan dokumen yang ditulisnya. Untuk membuat hubungan antara *author* dengan dokumen dalam model. Digunakan Kode 5.3.

```
1. doc_id_dict = dict(zip(doc_ids, range(le
   n(doc_ids))))
2. for a, a_doc_ids in author2doc.items():

3.     for i, doc_id in enumerate(a_doc_ids
   ):
4.         author2doc[a][i] = doc_id_dict[d
   oc_id]
5.
```

Kode 5.3 Mapping *author2doc*

Setelah seluruh data dimuat dan dihubungkan antara author dengan dokumen yang ditulisnya. Selanjutnya data list dokumen akan dilakukan proses pembersihan teks. Sedangkan untuk data list *author* tidak dilakukan pembersihan data karena hanya berisih nama *author* dan *id* dokumen yang ditulis.

5.3 Pra-proses

Tahap ini dilakukan agar data yang digunakan sesuai dengan data yang dibutuhkan model. Terdapat 5 tahapan dalam pra-

proses, yaitu *case folding*, *data cleaning*, *stopword removal*, menghapus kata dengan kode ascii lebih dari 128, dan menghapus kata yang tidak ada dalam *word2vec* model.

5.3.1 Pendefinisian *Stopword*

Sebelum melakukan penghapusan kata yang ada dalam *stopword*. Langkah pertama yang dilakukan adalah mendefinisikan kata apa saja yang termasuk kedalam *stopword*. Untuk mendifinisikan kata yang termasuk kedalam stopword, menggunakan *library nltk.corpus*. Implementasi proses pendefinisian *stopword* sesuai dengan Kode 5.4.

```
1. from nltk.corpus import stopwords
2.
3. stops = set(corpus.stopwords.words(filei
   ds='english'))
4.
```

Kode 5.4 Pendefinisian *Stopwords*

5.3.2 Pendefinisian kata dalam *word2vec* model

Sebelum melakukan penghapusan kata yang tidak ada dalam *word2vec* model, langkah pertama adalah mengunduh file *word2vec*. *file* yang telah terunduh lalu disimpan menggunakan *library genism.models.keyedvectors.* langkah selanjutnya untuk mendefinisikan kata yang terdapat dalam *word2vec* model sesuai dengan Kode 5.5.

```
1. from gensim.models import KeyedVectors
2.
3. wvmodel = gensim.models.KeyedVectors.loa
   d_word2vec_format("E:/andhika/glove.6B.5
   0d.word2vec.txt", binary=False)
4.
5. wv_vocab = set(wvmodel.vocab.keys())
6.
```

Kode 5.5 Pendefinisian *word2vec* model

5.3.3 *Case Folding* dan Penghapusan Kata

Dalam melakukan tahap penghapusan kata yang tidak diperlukan, dilakukan menggunakan kode *python 'filter'*. Data yang dibersihkan adalah data dengan format *txt* yang dimuat menjadi satu kedalam *list* dokumen. Seluruh hasil pra-proses dari *case folding* hingga penghapusan kata yang tidak dibutuhkan akan disimpan kedalam variable '*docs*'. Implementasi proses pembersihan data sesuai dengan Kode 5.6.

```
1. docs = []
2. doc_path = "E:/andhika/data/test/"
3. for folder in os.listdir(doc_path)[0:]:

4.     for doc in os.listdir(doc_path + fol
   der):
5.         with open(doc_path + folder + "/
   " + doc, 'r', encoding="utf8") as f:
6.             txt = f.read().split()
7.             txt = map(lambda x: x.lower(
   ), txt)  # Lowercasing each word
8.             txt = filter(lambda word: [l
   etter for letter in word if ord(letter)
   < 128 ], txt)  # Checking each word for
   ascci error
9.             txt = filter(lambda x: x not
    in stops, txt)  # Removing stop words
10.             txt = filter(lambda x: x.isa
   lpha(), txt)  # Removing non-
   letter words (eg numbers and symbols)
11.             txt = filter(lambda x: len(x
   ) > 2, txt)  # removing super short word
   s and single letters
12.             txt = filter(lambda x: x in
   wv_vocab, txt)
13.             txt = ' '.join(txt)
14.             docs.append(txt)
15.
```

Kode 5.6 Case Folding

5.3.4 Tokenisasi

Pada tahapan ini dilakukan tokenisasi atau mengubah kalimat menjadi kata dari setiap dokumen yang tersimpan pada variabel '*docs*'. Untuk melakukan tokenisasi pada penelitian ini menggunakan *library nltk.tokenize* dan disimpan pada variabel token. Tahapan ini dapat dilihat pada Kode 5.7

```
1. from nltk.tokenize import word_tokenize

2.
3. token = []
4. for a in docs:
5.     nltk_tokens = nltk.word_tokenize(a)

6.     token.append(nltk_tokens

7.
```

Kode 5.7 Tokenisasi

5.4 Pemodelan *Author-Topic* menggunakan *Gaussian LDA*

Tahap ini melakukan pemodelan *Author-Topic* menggunakan metode *Gaussian Latent Dirichlet Allocation.* Proses pemodelan dilakukan dengan memanfaatkan model *word2vec* yang telah didefinisika. Sebelum melakukan tahapan pemodelan, langkah pertama yang dilakukan adalah iniisiasi variable yang didefinisikan pada fungi *'__init__'* untuk menyimpan *corpus, dictionary, dan parameter-parameter lainnya.* Inisiasi variabel ditunjukkan pada Kode 5.8

```
1. def __init__(self, num_topics, corpus, a
   uthor2doc, doc2author=None, word_vector_
   filepath=None,
2.                  word_vector_model=None,
    alpha=0.2, outputfile=None, preprocess=
   False, run_name=str(1), minimum_probabil
   ity = 0.01):
3.         self.doc_topic_CT = None
4.         self.corpus = corpus
5.         self.priors = None
```

```
6.          self.word_vecs = {}
7.          self.numtopics = num_topics
8.          self.vocab = set([])
9.          self.topic_params = defaultdict(
   dict)
10.         self.wordvecFP = word_vector_fil
   epath
11.         self.word_vec_size = None
12.         self.alpha = alpha
13.         self.wvmodel = word_vector_model

14.         self.word_topics = {}
15.         self.output_file_name = outputfi
   le
16.         self.preprocess = preprocess
17.         self.run_name = run_name
18.         self.dokumen_topik = []
19.         self.author_top = []
20.         self.author2doc = author2doc
21.         self.dokumenuji_topik = []
22.         self.minimum_probability = minim
   um_probability
```

Kode 5.8 Init Variabel

5.4.1 Pemrosesan *Corpus* dan *dictionary*

Pada tahap ini dilakukan pemrosesan *corpus* dan *dictionary* dari setiap dokumen yang telah disimpan pada variabel '*docs*'. Lalu setiap kata yang terdapat pada dokumen akan dikelompokkan secara acak kedalam masing-masing topik sesuai jumlah topik yang ditentukan. Hasil dari pemrosesan *corpus* akan disimpan kedalam variabel *self.corpus* yang berisi kata dengan *integer* topiknya. Sedangkan untuk pemrosesan *dictionary* akan disimpan kedalam variabel *self.vocab* yang berisi jenis kata unik. Tahapan ini dilakukan sesuai dengan Kode 5.9

```
1.  def process_corpus(self, documents):
2.    if not self.preprocess:
3.      temp_corpus = defaultdict(dict)
4.      random.shuffle(documents)
5.      for index, doc in enumerate(documents):
```

```
            words = doc.split()
            temp_corpus[index]['words'] = words
            temp_corpus[index]['topics'] = np.empty(len(words))  # Random topic assign
            for word in words:
                self.vocab.add(word)
        self.corpus = temp_corpus
        print ("Done processing corpus with {} documents".format(len(documents)))

    else: # Docs are tokenized and such, just add it into class
        temp_corpus = defaultdict(dict)
        for idx, doc in enumerate(documents):
            temp_corpus[idx]["words"] = doc
            temp_corpus[idx]["topics"] = np.empty(len(doc))
            for word in words:
                self.vocab.add(word)
        self.corpus = temp_corpus
```

Kode 5.9 Corpus dan Dictionary *GLDA*

5.4.2 Pemrosesan Vektor Kata

Pada tahap ini dilakukan pengubahan setiap kata unik yang ada pada variabel *self.vocab* kedalam bentuk vektor kata yang ada pada *word2vec* model. Vektor kata yang terpadat pada *word2vec* disimpan kedalam variabel *self.wordvecFP*, lalu diambil nilai vektor kata menggunakan *library genism.models* dengan fungsi *KeyedVectors* dan disimpan kedalam variabel *self.wvmodel*.

Konversi kata menjadi vektor dilakukan dengan cara mengambil kata yang ada pada variabel self.vocab. Kata yang diambil tersebut lalu diubah menjadi vektor kata dengan vektor kata yang sama pada self.wvmodel. Jika terdapat kata yang tidak memiliki vektor kata, maka kata tersebut diubah kedalam vektor kata ‘unk’. Hasil dari pengubahan vektor kata tersebut disimpan kedalam variabel self.word_vecs. Proses pada tahapan ini sesuai dengan Kode 5.10

```
def process_wordvectors(self, filepath=None):
if filepath:
print ("Processing word-vectors, this takes a moment")
self.wvmodel = KeyedVectors.load_word2vec_format(fname=filepath, limit=200000)
useable_vocab = 0
unusable_vocab = 0
self.word_vec_size = self.wvmodel.vector_size


for word in self.vocab:
try:
self.word_vecs[word] = self.wvmodel[word] #match between word on corpus and word embedding
useable_vocab += 1
except KeyError:
self.word_vecs[word] = self.wvmodel['unk']
unusable_vocab += 1

print ("There are {0} words that could be converted to word vectors in your corpus \n" \
                "There are {1} words that could NOT be converted to word vectors".format(useable_vocab,unusable_vocab))
else:
useable_vocab = 0
unusable_vocab = 0
self.word_vec_size = self.wvmodel.vector_size

for word in self.vocab:
try:
self.word_vecs[word] = self.wvmodel[word]
useable_vocab += 1
except KeyError:
```

```
30. self.word_vecs[word] = self.wvmodel['unk
    ']
31. unusable_vocab += 1
32.
33. print ("There are {0} words that could b
    e converted to word vectors in your corp
    us \n" \
34.                 "There are {1} words that
    could NOT be converted to word vectors".
    format(useable_vocab,unusable_vocab))
```

Kode 5.10 Pemrosesan Vektor Kata

5.4.3 Pembentukan *Author-Topik* Model

Pada tahap pembentukan model dilakukan dengan dua tahapan, yaitu inisiasi parameter dan *sampling*. Inisiasi parameter dilakukan dengan memanggil fungsi *'init'*. Parameter yang dimaksud adalah *prior mean*, *prior co-variance*, *sample means*, *sample covariances*, dan *document-topic count*. Parameter *prior* berfungsi sebagai parameter perhitungan sebelum *gibs sampling*. Sedangkan parameter *mean* merupakan parameter yang digunakan sebagai perhitungan pada saat dilakukan *sampling*. Untuk parameter *document-topic count* digunakan untuk perhitungan dan melihat banyaknya persebaran kata pada setiap topik. Tahapan inisiasi parameter dilakukan sesuai dengan Kode 5.11

```
1.  mu_0 = np.zeros(self.word_vec_size)
2.  count = 0
3.  for docID in self.corpus.keys():
4.
5.  for i, word in enumerate(self.corpus[docID]['words']):
6.  self.corpus[docID]['topics'][i]=self.word_topics[word]
7.  mu_0 += self.word_vecs[word]
8.  count += 1
9.
10. # Prior Mean
11. self.priors.mu = mu_0 / float(count)
12.
13. # Prior co-variance
14. self.priors.psi = .01 * np.identity(self.word_vec_size)
```

```
15.
16. # Sample means
17. for k in range(self.numtopics):
18. self.topic_params[k]["Topic Sum"] = np.zeros(self.word_
    vec_size)
19. self.topic_params[k]["Topic Mean"] = centroids[k]
20. self.topic_params[k]["Topic Cov"] = np.zeros((self.word_
    vec_size, self.word_vec_size))
21.
22. # Sample co-variances and document-topic counts
23. co_variances = [np.zeros((self.word_vec_size, self.word_v
    ec_size)) for _ in range(self.numtopics)]
24. for docID in self.corpus.keys():
25. for topic, word in zip(self.corpus[docID]['topics'], self.co
    rpus[docID]['words']):
26. topic = int(topic)
27. wv = self.word_vecs[word]
28. sample_mu = self.topic_params[topic]["Topic Mean"]
29. # Init Document-Topic counts
30. self.doc_topic_CT[docID, topic] += 1.
31. # Sum of topic vectors
32. self.topic_params[topic]['Topic Sum'] += wv
33. #+ self.priors.psi(vk) --- ck (sample covariance)
34. co_variances[topic] += np.outer(wv - sample_mu, wv - sa
    mple_mu)
35.
36. #   Normalize the sample co-variance
37. for k in range(self.numtopics):
38. self.topic_params[k]["Topic Cov"] = (co_variances[k] / (n
    p.sum(self.doc_topic_CT[:, k]) - 1.)) + self.priors.psi

39.
```

Kode 5.11 Inisiasi Parameter

Setelah inisiasi dilakukan, langkah selanjutnya adalah *sampling* untuk mengelompokkan tiap kata yang ada pada bariabel *self*.corpus ke dalam topik dengan menggunakan metode *Gibbs Sampling*. Langkah pertama yang dilakukan dalam *sampling* adalah mendeklarasikan variabel yang digunakan untuk menyimpan kumpulan dokumen, kata yang terkumpul dari

dokumen, dan topik awal yang sudah didefinisikan secara random sebelumnya. Inisiasi variabel ditampilkan pada Kode 5.12

```
for docID in self.corpus.keys():
for idx in range(len(self.corpus[docID]['words'])):
word = self.corpus[docID]['words'][idx]
current_topic = self.corpus[docID]['topics'][idx]

```

Kode 5.12 Memanggil Dokumen

Lalu dilakukan perhitungan ulang dari nilai parameter *mean* dan *covariance*. Langkah pertama yang dilakukan adalah memperbarui jumlah distribusi kata dan vektor kata pada setiap topik. Hal tersebut dilakukan dengan memanggil fungsi *update_document_topic_counts* dengan operasi masukkan negatif dan positif. Jika operasi negatif, maka jumlah kata pada variabel *doc_topic_CT* dikurangi satu dan jumlah vektor kata pada variabel *self.topic_params* dikurangi dengan vektor kata tersebut. Kalau operasi masukkan positif, maka akan berkebalikan dengan apa yang dilakukan saat operasi masukkan negatif. Tahap memperbarui jumlah distribusi kata dapat dilihat pada Kode 5.13.

```
self.recalculate_topic_params(word, int(current_topic), i
nt(docID), "-")
if UPDATE_COUNT:
 self.update_document_topic_counts(word, topic, docID,
operation)

def update_document_topic_counts(self, word, topic, doc
ID, operation):
 if operation == "-":
  self.topic_params[topic]["Topic Sum"] -
= self.word_vecs[word] #jumlah vektor kata dalam topik
minus(-) current_kata
  self.doc_topic_CT[int(docID), int(topic)]-
= 1. #jumlah kata dalam topik per dokumen dikurangi

```

```
  if operation == "+":
    self.topic_params[topic]["Topic Sum"] += self.word_vecs[word]
    self.doc_topic_CT[docID, topic] += 1.

```

Kode 5.13 Memperbarui Jumlah Distribusi Kata

Selanjutnya dilakukan perhitungan ulang terhadap nilai *topic mean* dan *topic covariance*. Tahap tersebut dapat dilakukan dengan memanggil fungsi *recalculate_topic_params* dengan operasi masukkan negatif dan positif. Jika operasi negatif, maka nilai *topic covariance* dikurangi dengan nilai *recursive downdate*. Kode perhitungan ulang untuk operasi negatif dapat dilihat pada Kode 5.14

```
self.recalculate_topic_params(word, int(current_topic), int(docID), "-")
if operation == "-":
  if UPDATE_DISTS:
    wv = self.word_vecs[word]
    mu = self.topic_params[topic]["Topic Mean"]

    # Get rank-1 matrix from point
    centered = wv - mu
    # Scale for recursive downdate
    centered *= np.sqrt((kappa_k + 1.) / kappa_k)
    self.topic_params[topic]["Topic Cov"] -= np.outer(centered, centered)


    # Correct the mean for the removed point
    sample_mean_K = self.topic_sample_mean(topic, topic_count)
    topic_mean = ((self.priors.kappa * self.priors.mu) + (topic_count * sample_mean_K)) / kappa_k  # Mu_k

```

Kode 5.14 Perhitungan Parameter (-)

Sedangkan, jika operasi masukkan positif. Maka, nilai *topic covariance* ditambah dengan nilai *recursive downdate*. Kode perhitungan ulang untuk operasi masukkan positif dapat dilihat pada Kode 5.15

```
else:
 sample_mean_K = self.topic_sample_mean(topic, topic_count)  # V-Bar_k
 topic_mean = ((self.priors.kappa * self.priors.mu) + (topic_count * sample_mean_K)) / kappa_k  # Mu_k

 if UPDATE_DISTS:
  centered = (self.word_vecs[word] - topic_mean)
  centered *= np.sqrt(kappa_k / (kappa_k - 1.))  # Scale for recursive update
  self.topic_params[topic]["Topic Cov"] += np.outer(centered, centered)

```

Kode 5.15 Perhitungan Parameter (+)

Kemudian dilakukan tahapan *gibbs sampling*. Pada tahapan ini dilakukan penghitungan dari nilai variabel *log_posterior* dari semua topik yang telah ditentukan. Hasil dari nilai variabel *log_posterior* berbentuk array dari probabilitas kata untuk setiap topik. Penghitungan nilai *log_posterior* sesuai dengan Kode 5.16.

```
log_posterior = np.zeros(self.numtopics)
for k in range(self.numtopics):
 log_pdf = self.draw_new_wt_assgns(word, k)
 # Count of topic in doc, Ndk
 Nkd = self.doc_topic_CT[docID, k]
 log_posterior[k] = np.real(np.log(Nkd + self.alpha) + log_pdf)

```

Kode 5.16 Perhitungan *Log_Posterior*

Niilai dari variabel *log_pdf* yang terdapat pada Kode **5.13** didapatkan dari perhitungan nilai *log of the probability density*

yang didefinisikan pada fungsi *draw_new_wt_assgns*. Kode perhitungannya sesuai dengan Kode 5.17

```
def draw_new_wt_assgns(self, word, topic, new_doc=Fals
e, wvmodel=None):
 cov_det = self.topic_params[topic]["Chol Det"] # covaria
nce determinan
 Nk = self.topic_params[topic]["Topic Count"] # count of
topic
 try:
  centered = self.word_vecs[word] - self.topic_params[to
pic]["Topic Mean"]
 except KeyError:
  self.word_vecs[word] = self.wvmodel['unk']
  centered = self.word_vecs[word] - self.topic_params[to
pic]["Topic Mean"]
 d = self.word_vec_size  # dimensionality of word vector
 kappa_k = self.topic_params[topic]["Topic Kappa"]

 scaleT = np.sqrt((kappa_k + 1.) / kappa_k * (self.priors.n
u - d + 1.))  # Covariance = chol / sqrt(scaleT)
 nu = self.priors.nu + Nk - d + 1.

 # Covariance Inverse
 cov_inv = self.topic_params[topic]["Topic Inv"]
 cov_inv = centered.T.dot(cov_inv).dot(centered)
 cov_inv *= scaleT

 cov_det = self.topic_params[topic]["Topic Det"]

 # Gaussian
 a = gammaln((nu + d) / 2.)
 b =  (gammaln(nu / 2.) + (d / 2.) * (np.log(nu) + np.log(p
i)) + (0.5 * cov_det) + ((nu + d) / 2.) * np.log(1. + cov_inv/
nu))
 # Log Multivariate T - PDF
return a-b

```

Kode 5.17 ***Log of the Probability Density***

Kemudian tiap kata pada variabel *self.corpus* yang telah melalui tahapan *gibbs sampling* diproses untuk mencari nilai probability tertinggi yang dapat dilihat dari variable *log_posterior*. Kata tersebut nantinya akan dimasukkan kedalam topik dengan nilai probability tertinggi. Setelah tahapan selesai dilakukan, maka jumlah topik distribusi pada tiap topik akan diperbarui, dan nilai parameter seluruh vektor kata pada tiap topik baru akan diperbarui. Tahapan ini sesuai dengan Kode 5.18

```
1.  ASSIGN_NEW_TOPICS = True
2.  MULTINOMIAL_TOPIC_SELECTION = True
3.
4.  max_log_posterior = np.max(log_posterior)
5.  log_posterior -= max_log_posterior
6.  normalized_post = np.exp(log_posterior - np.log(np.sum(
    np.exp(log_posterior))))
7.
8.  if MULTINOMIAL_TOPIC_SELECTION:
9.    new_topic = np.argmax(np.random.multinomial(1, pval
    s=normalized_post))
10. else:
11.   new_topic = np.argmax(normalized_post)
12.
13. if not ASSIGN_NEW_TOPICS:
14.   new_topic = current_topic
15.
16. self.corpus[docID]['topics'][idx] = new_topic
17. self.recalculate_topic_params(word, new_topic, docID, "+
    ")
18.
```

Kode 5.18 Topik Baru

5.4.4 Eksperimen Pemodelan Topik

Pada tahapan ini dilakukan eksperimen untuk mencari model yang terbaik. Hal tersebut dilakukan dengan mengubah parameter masukkan seperti jumlah iterasi dan jumlah topik.

5.4.4.1 Penentuan Jumlah Iterasi

Eksperimen penentuan jumlah iterasi dilakukan terlebih dahulu sebelum menentukan jumlah topik. Eksperimen dilakukan sebanyak lima kali percobaan dengan input parameter 1, 2, 3, 4, dan 5. Nilai yang digunakan sebagai penentu jumlah iterasi adalah nilai perplexity dari tiap iterasi yang dilakukan. Nilai perplexity didapatkan dari nilai probability dari tiap kata yang ada pada *self.corpus* tuntuk setiap topik yang ada. Tahapan untuk mencari nilai probability dari tiap kata untuk setiap topik dilakukan sesuai dengan Kode 5.19

```
1.  def probability_perplex(self, iteration):
2.   with open('E:/andhika/hasil/prob/' + str(self.numtopics) + 'topic_' + str(iteration) + 'iteration' +  '.txt', 'w',encoding="utf8") as final_prob:
3.
4.   counter=0
5.   topik = []
6.   doktopik = []
7.   c = Counter(self.word_topics.values())
8.   for docID in self.corpus.keys():
9.    counter+=1
10.   doc_topik = []
11.   doc_topics = []
12.   count=0
13.   for word in self.corpus[docID]['words']:
14.    count+=1
15.    posterior = []
16.
17.    for k in range(self.numtopics):
18.     prob = self.draw_new_wt_assgns(word, k, wvmodel=self.wvmodel, new_doc=True) + log(self.alpha + c[k])
19.     posterior.append(prob)
20.    posterior /= np.sum(posterior)
21.    doc_topics.append(posterior)
22.    for m in (posterior):
23.     final_prob.write("{2},{0},{1},{3}\n".format(word,m,docID,np.argmin(posterior)))
24.    for t in zip(*(doc_topics)):
25.     doc_topik.append((np.sum(t)))
```

```
26.    doc_topik /= np.sum(doc_topik)
27.    topik.append(doc_topik)
28.    del(doc_topics)
29.    self.dokumen_topik.append(doc_topik)
30.
```

Kode 5.19 Nilai Probability

Setelah nilai probabilitas didapatkan, nilai dari probabilitas tiap kata disimpan kedalam sebuah file dokumen. Lalu langkah selanjutnya adalah menghitung nilai perplexity dari file dokumen tersebut. Tahapan menghitung nilai perplexity sesuai dengna Kode 5.20.

```
1.  import numpy as np
2.  import os
3.  i = 500
4.  iterasi = 8
5.  for a in range(iterasi):
6.    folder = 'E:/andhika/hasil/prob/'
7.    file = str(i) + 'topic_' + str(a) + 'iteration' + '.txt'
8.    doc_path = folder + 'sampah/' + file
9.    with open(folder+file, 'r', encoding="utf8") as fi:
10.     a = 0 #total nilai probability
11.     z = 0 #jumlah kata
12.
13.     for line in fi:
14.       px= float(line.split(',')[2])
15.       z+=1
16.       nilai = np.log10(px)
17.       if str(nilai) != 'nan':
18.         a += nilai * px
19.
20.     c=-(a/z)
21.     d=np.exp(c)
22.     print(d)
23.
```

Kode 5.20 Nilai *Perplexity*

5.4.4.2 Penentuan Jumlah Topik

Setelah mendapatkan nilai parameter yang terbaik, selanjutnya mencari jumlah topik yang sesuai dengan model. Eksperimen ini dilakukan sebanyak 12 kali percobaan dengan input parameter 25, 50, 75, 100, 125, 150, 175, 200, 225, 250, 275, 300. Penentuan jumlah topik berguna untuk penentuan distribusi kata pada tiap topik. Semakin kecil nilai perplexity, maka semakin baik model tersebut atau semakin bagus persebaran kata pada tiap topiknya. Tahapan penentuan jumlah topik dilakukan sesuai dengan Kode **5.18** dan **5.19**, yaitu mencari nilai probability tiap kata dan menghitung nilai perplexitynya

5.4.4.3 Menyimpan Model

Setelah mendapatkan model yang terbaik, maka model tersebut akan disimpan dengan menggunakan *library joblib.dump*. Penyimpanan ini dilakukan agar tidak perlu melakukan training model lagi saat ingin melakukan pengujian maupun melihat parameter yang terdapat didalamnya. Tahapan menyimpan model dilakukan sesuai dengan Kode 5.21

```
1. #Menyimpan model
2. #parameter (Nama model, Nama file)
3. joblib.dump(lda, 'modelglda50')
4.
```

Kode 5.21 Menyimpan Model

5.5 Pemodelan *Author-Topic* Menggunakan *LDA*

Tahap pemodelan *Author-Topic* dengan menggunakan metode *Latent Dirichlet Allocation* merupakan tahapan membentuk model menggunakan *library genism*. Selain itu, juga dilakukan eksperimen untuk mendapatkan model yang terbaik.

5.5.1 Pembentukan Corpus dan Dictionary

Sebelum membentu *corpus* dan *dictionary*, langkah pertama yang dilakukan adalah melakukan tokenisasi dari setiap dokumen dan dimasukkan kedalam variabel token. Tahapan

melakukan tokenisasi dilakukan menggunakan *library nltk* dan sesuai dengan Kode 5.22.

```
from nltk.tokenize import word_tokenize

token = []
for a in docs:
 nltk_tokens = nltk.word_tokenize(a)
 token.append(nltk_tokens)

```

Kode 5.22 Tokenisasi LDA

Variabel token adalah masukkan untuk pembentukan *dictionary*. Pembentukan *dictionary* dilakukan dengan menggunakan *library gensim.dictionary*. Hasil dari *dictionary* adalah jenis kata unik dengan *id* dari kata tersebut. Tahapan untuk membuat *dictionary* sesuai dengan Kode 5.23

```
dictionary = Dictionary(token)

_ = dictionary[0]  # This sort of "initializes" dictionary.id
2token.

```

Kode 5.23 Dictionary *LDA*

Setelah *dictionary* terbentuk, maka dilakukan pembentukan *corpus* yang dilakukan dengan masukkan *dictionary*. Hasil dari pembentukan *corpus* ini adalah mengubah unik kata menjadi *bag of word* untuk setiap kata. Tahapan pembentukan corpus dilakukan sesuai dengan Kode 5.24

```
corpus = [dictionary.doc2bow(doc) for do
c in token]

```

Kode 5.24 *Corpus LDA*

5.5.2 Pembentukan *Author-Topik* Model

Tahap awal dalam pemodelan *author-topik* adalah membuat model. Untuk membaut model dilakukan dengan menggunakan

library gensim.atmmodel. Beberapa parameter penting yang harus diperhatikan dalam pembentukan model adalah jumlah topik dan iterasi. Kedua parameter tersebut nantinya digunakan untuk melakukan eksperimen mencari model yang terbaik. Tahap pembentukan *author-topik* model sesuai dengan Kode 5.25

```
1. model = AuthorTopicModel(corpus=corpus,
   num_topics=125, id2word=dictionary.id2to
   ken, author2doc=author2doc, passes=75, e
   val_every=10)
2.
```

Kode 5.25 Modeling *Author-Topic LDA*

5.5.3 Eksperimen Pemodel *Author-Topik* menggunakan *LDA*

Pada tahapan ini dilakukan eksperimen untuk mencari model yang terbaik. Hal tersebut dilakukan dengan mengubah parameter masukkan seperti jumlah iterasi dan jumlah topik. Berdasarkan hal ini, maka eksperimen dilakukan dengan tahapan penentuan jumlah passes dan penentuan jumlah topik.

5.5.3.1 Penentuan Jumlah Passes

Passes atau iterasi merupakan perulangan yang dilakukan oleh *LDA* untuk melakukan pembelajaran pada setiap passes. Maka dari itu penentuan jumlah passes sangat penting untuk mendapatkan perulangan yang paling tepat sehingga mendapatkan model dengan pembelajaran yang baik. Jumlah passes dapat memengaruhi distribusi kata pada setiap topik, jumlah passes yang terlalu sedikit dapat menyebabkan distribusi kata-kata pada setiap topik masih terlihat berbeda, begitu pula dengan jumlah passes yang terlalu banyak.

Eksperimen untuk mendapatkan jumlah passes yang terbaik dilakukan sebanyak delapan kali dengan jumlah topik yang berbeda-beda. Parameter yang digunakan untuk eksperimen adalah 5, 10, 15, 20, 25, 50, 75, dan 100 Passes. Untuk parameter jumlah topik yang digunakan adalah 25, 50 ,75, 100, 125, 150, 175, 200, dan 225. Hasil dari eksperimen tersebut adalah nilai perplexity dari tiap iterasi. Hasil yang didapatkan

akan dianalisa dengan melihat tingkat kestabilan nilai perplexitynya.

Untuk mendapatkan nilai perplexity di setiap passes, menggunakan *library logging* sehingga nilai perplexity akan tercatat dan muncul pada log. Tahap eksperimen jumlah passes dan mendapatkan nilai perplexity sesuai dengan Kode 5.26

```
1. import logging
2. logging.basicConfig(format='%(asctime)s
   : %(levelname)s : %(message)s', level=lo
   gging.INFO)
3.
4.
5. %time model = AuthorTopicModel(corpus=co
   rpus, num_topics=125, id2word=dictionary
   .id2token, author2doc=author2doc, passes
   =75, eval_every=10)
6.
```

Kode 5.26 Nilai *Perplexity* dari *Passes*

Variabel *%time* pada Kode **5.23** digunakan untuk melihat durasi dari pembuatan model. Seluruh hasil perplexity yang didapatkan dari setiap passes akan divisualisasikan agar dapat dianalisa tren kestabilan passes.

5.5.3.2 Penentuan Jumlah Topik

Setelah mendapatkan jumlah passes yang sudah stabil, dilanjutkan dengan eksperimen menentukan jumlah topik. Penentuan jumlah topik juga dirasa penting, karena dapat mempengaruhi distribusi kata pada setiap topik. Jika topik terlalu sedikit, maka distribusi kata yang muncul akan cenderung ke kata yang sering muncul pada dokumen tersebut. Sedangkan, kalau topik terlalu banyak, maka akan membuat distribusi kata pada setiap topik akan terlihat berjauhan.

Eksperimen untuk mendapatkan jumlah topik yang terbaik dilakukan sebanyak sembilan kali dengan jumlah passes yang sudah stabil. Parameter yang digunakan untuk eksperimen adalah 25, 50, 75, 100, 125, 150, 175, 200, 225.

Untuk menentukan jumlah topik terbaik berbeda dengan penentuan jumlah passes. Pada penentuan jumlah topik dilakukan perulangan training sebanyak sepuluh kali untuk setiap passesnya. Lalu dihitung rata-rata dan standard deviasi dari nilai perplexity terakhir untuk setiap passes. Tahapan penentuan jumlah topik dilakukan sesuai dengan Kode 5.27

```
1. import logging
2. logging.basicConfig(format='%(asctime)s
   : %(levelname)s : %(message)s', level=lo
   gging.INFO)
3.
4.
5. %time model = AuthorTopicModel(corpus=co
   rpus, num_topics=125, id2word=dictionary
   .id2token, author2doc=author2doc, passes
   =75, eval_every=10)
6.
```

Kode 5.27 Nilai *Perplexity* dari Jumlah Topik

5.5.3.3 Menyimpan Model

Setelah mendapatkan model yang terbaik dari kedua eskperimen yang dilakukan, maka model tersebut akan disimpan dengan menggunakan *library joblib.dump*. Penyimpanan ini dilakukan agar tidak perlu melakukan training model lagi saat ingin melakukan pengujian maupun melihat parameter yang terdapat didalamnya. Tahapan menyimpan model dilakukan sesuai dengan Kode 5.28

```
1. #Menyimpan model
2. #parameter (Nama model, Nama file)
3. joblib.dump(lda, 'modelglda125')
4.
```

Kode 5.28 Menyimpan Model

5.6 Validasi Model Author-Topic *Gaussian LDA*

Setelah didapatkan model yang terbaik berdasarkan nilai perplexity dari hasil eksperimen, selanjutnya dilakukan validasi

model berdasarkan nilai *topic coherence*. Untuk menghitung nilai *topic coherence* pada *Gaussian LDA* menggunakan metode *pointwise mutual information(PMI).*

Metode *PMI* dilakukan dengan cara mengambil lima kata yang memiliki probability terbaik pada setiap topik. Dari lima kata tersebut dihitung dengan membandingkan probabilitas kemunculan tiap dua kata secara bersamaan terhadap nilai probabilitas kemunculan kata pertama dengan kata kedua pada setiap dokumen. Tahapan ini dilakukan sesuai dengan Kode 5.29

```
countup = 0
while countup < 175:
 cb = zxc[countup]
 cb2 = zxc[countup]
 i=0
 rataan = 0
 for katas in cb:
   i+=1
   for katass in cb2[i:]:
     if katas!=katass:
      word1 = katas
      word2 =  katass
      total_doc = 0
      total_word = 0
      tf_word1 = 0
      tf_word2 = 0
      total_coocurence = 0
      for doc in corpus:
       find_word1 = 0
       find_word2 = 0
       token = doc.split()
       if word1 in token:
        find_word1 = 1
        tf_word1 += 1
       if word2 in token:
        find_word2 = 1
        tf_word2 += 1
       if(find_word1 == 1 and find_word2
  == 1):
```

```
29.          total_coocurence += 1
30.
31.         total_doc += 1
32.         total_word += len(token)
33.
34.        px = tf_word1/total_word
35.        py = tf_word2/total_word
36.        pxy = total_coocurence/total_doc
37.        try:
38.         pmi = (pxy / (px * py))
39.        except ZeroDivisionError:
40.         pmi = 0
41.        pmo = np.log10(pmi)
42.        rataan+=pmo
43.  print(rataan/10)
44.  countup+=1
45.
```

Kode 5.29 Validasi Model *GLDA*

Setelah seluruh nilai *topic coherence* didapatkan, lalu dihitung rata-rata dari nilai tersebut pada setiap topik untuk mengetahui nilai *topic coherence* pada masing-masing topik. Nilai *topic coherence* untuk model dihitung dari rata-rata nilai *topic coherence* dari masing-masing topik.

5.7 Validasi Model *Author-Topic LDA*

Setelah didapatkan model yang terbaik berdasarkan nilai perplexity dari hasil eksperimen, selanjutnya dilakukan validasi model berdasarkan nilai *topic coherence*. Untuk menghitung nilai *topic coherence* dari model *author-topic LDA* menggunakan metode yang sama dengan model sebelumnya yaitu *pointwise mutual information*. Terdapat dua tahapan untuk melakukannya, yaitu memanggil top topics dan menghitung rata-rata coherence score.

Sebelum melakukan validasi topik, dilakukan tahap memuat model yang sudah disimpan sebelumnya, Tahapan untuk memuat model yang sudah terseimpan sebelumnya sesuai dengan Kode 5.30

```
#nama model baru = joblib.load('nama file model')

atm = joblib.load('modelatm175')

```

Kode 5.30 Memuat Model

5.7.1 Memanggil *Top Topics*

Pada tahap ini dilakukan pemanggilan topik terbaik dari setiap model. Pemanggilan ini dilakukan untuk mendapatkan distribusi kata pada seluruh topik yang ada pada model. Perintah yang digunakan untuk memanggil *top topics* pada model tertera pada Kode 5.31

```
top_topics = atm.top_topics(atm.corpus)

pprint(top_topics)

```

Kode 5.31 Memanggil Top Topics

5.7.2 Rata-rata *Coherence Score*

Setelah mengetahui nilai *topic coherence* dari tiap topik pada sebuah model, langkah selanjutnya adalah mencatat 5 kata yang memiliki distribusi terbaik pada setiap model. Lalu setiap kata dihitung nilai *topic coherence* menggunakan metode *pointwise mutual information.* Langkah untuk melakukan penghitungan *topic coherence* dengan menggunakan metode *pointwise mutual information* sama dengan langkah pada validasi model *author-topic* menggunakan *Gaussian LDA.* Implementasi perhitungan rata-rata sesuai dengan Kode 5.32

```
zxc = []
with open('E:/cobaan.txt','r') as f:
 a = f.read().splitlines()
  for b in a:
   zxc.append(b.split())
countup = 0
while countup < 175:
 cb = zxc[countup]
```

```
cb2 = zxc[countup]
i=0
rataan = 0
for katas in cb:
 i+=1
 for katass in cb2[i:]:
  if katas!=katass:
   word1 = katas
   word2 =  katass
   total_doc = 0
   total_word = 0
   tf_word1 = 0
   tf_word2 = 0
   total_coocurence = 0
   for doc in docs:
    find_word1 = 0
    find_word2 = 0
    token = doc.split()
    if word1 in token:
     find_word1 = 1
     tf_word1 += 1
    if word2 in token:
     find_word2 = 1
     tf_word2 += 1
    if(find_word1 == 1 and find_word2 ==
 = 1):
     total_coocurence += 1
    total_doc += 1
    total_word += len(doc)
   px = tf_word1/total_word
   py = tf_word2/total_word
   pxy = total_coocurence/total_doc
   try:
    pmi = (pxy / (px * py))
   except ZeroDivisionError:
    pmi = 0
   pmo = np.log10(pmi)
   rataan+=pmo
print(rataan/10)
countup+=1
```

Kode 5.32 Validasi Model *LDA*

5.8 Pengujian Model *Author-Topic LDA* Dengan Menggunakan Nilai Similaritas Vektor Probabilitas

Tahapan ini dilakukan untuk menguji kemampuan prediksi dari model yang telah disimpan sebelumnya dalam mengetahui *author* yang membuat dokumen uji. Hal ini dilakukan dengan menghitung nilai similarity dari vektor probabilitas yang dimiliki *author* dengan dokumen menggunakan tiga metode, yaitu metode *Hellinger distance*, *cosine similarity*, dan *topic similarity*.

5.8.1 Memuat Data Pengujian

Tahap pertama yang dilakukan untuk menguji prediksi dari model adalah memuat data yang akan digunakan untuk pengujian. Data pengujian yang digunakan berjumlah 20 dengan syarat *author* pada data uji memiliki minimal satu dokumen *training* atau telah terdefinisi sebelumnya pada model. Tahap memuat data sesuai dengan Kode 5.33

```
1. docs_test = []
2. doc_path_test = "E:/andhika/data/testing
   /"
3. for folder in os.listdir(doc_path_test)[
   0:]:
4.   for doc in os.listdir(doc_path_test +
   folder):
5.   with open(doc_path_test + folder + "/"
    + doc, 'r', encoding="utf8") as f:
6.
```

Kode 5.33 Memuat Data Pengujian

Setelah data dimuat, lalu data dibersihkan sesuai dengan yang dilakukan pada saat pembersihan data *training*. Hasil dari tahap pembersihan data akan disimpan kedalam variabel '_'. Tahapan pembersihan data dilakukan sesuai dengan Kode 5.34

```
1. docs_test = []
2. with open(doc_path_test + folder + "/" +
    doc, 'r', encoding="utf8") as f:
3.   txt = f.read().split()
```

```
    # Lowercasing each word
    txt = map(lambda x: x.lower(), txt)
    # Checking each word for ascci error
    txt = filter(lambda word: [letter for letter in word if ord(letter) < 128 ], txt)
    # Removing stop words
    txt = filter(lambda x: x not in stops, txt)
    # Removing non-letter words (eg numbers and symbols)
    txt = filter(lambda x: x.isalpha(), txt)
    # removing super short words and single letters
    txt = filter(lambda x: len(x) > 2, txt)
    # removing words not in word2vec model
    txt = filter(lambda x: x in wv_vocab, txt)
    txt = ' '.join(txt)
    docs_test.append(txt)

```

Kode 5.34 Membersihkan Data Uji

5.8.2 Menghitung Probabilitas Kata ke dalam setiap topik pada dokumen uji

Setelah data selesai dibersihkan, selanjutnya dilakukan perhitungan probabilitas distribusi tiap kata ke dalam topik pada dokumen uji. Setiap kata akan memiliki nilai probabilitas untuk topik yang terdistribusi. Nilai probabilitas yang dimiliki tiap kata untuk setiap topik yang sama dijumlahkan dan dibagi dengan jumlah kata yang masuk kedalam topik tersebut. Setelah dilakukan proses tersebut, maka didapatkan probabilitas dokumen untuk setiap topik. Nilai probabilitas dari dokumen disimpan kedalam *array list* '*docs_vecs'* untuk memudahkan pemanggilan pada fungsi similaritas nanti. Tahapan ini dilakukan sesuai dengan Kode 5.35

```
doc_vecs = []
for data in docs_test:
 a = data.split()
 topics_sum = dict()
 topics_total = dict()
 topics_avg = dict()
 testkata = []
 for b in a:
  for (keys, values) in atm.id2word.items():
   if values == b:
    if values not in testkata:
     testkata.append(values)
     probTopics = atm.get_term_topics(keys)
     for (id_prob,val_prob) in probTopics:
      if id_prob not in topics_sum:
       topics_sum[id_prob] = val_prob
       topics_total[id_prob] = 1
      else:
       topics_sum[id_prob] = topics_sum[id_prob] + val_prob
       topics_total[id_prob] += 1
     topics_avg[id_prob] = topics_sum[id_prob] / topics_total[id_prob]
 doc_vecs.append(list(topics_avg.items()))

```

Kode 5.35 Probabilitas Kata

5.8.3 Menghitung Nilai Similaritas Vektor Probabilitas

Pada tahapan ini dilakukan penghitungan nilai similaritas dari vektor probabilitas yang dimiliki oleh *author* dan dokumen dengan menggunakan tiga metode. Untuk menghitung nilai similaritas vektor probabilitas menggunakan *Hellinger distance* dilakukan dengan *library genism.matutils hellinger*. Sebelum dilakukan penghitungan, nilai vektor probabilitas dari *author* dan dokumen diubah terlebih dahulu kedalam bentuk *array*

dengan menggunakan *library matutil.sparse2full*. Implementasi *Helinger distance* sesuai dengan Kode 5.36

```
from gensim import matutils
from sklearn.metrics.pairwise import cosine_similarity
from scipy import spatial

author_vecs = [atm.get_author_topics(author) for author in atm.id2author.values()]

# Get similarity between two vectors
def similarity(vec1, vec2):
    a = matutils.sparse2full(vec1, atm.num_topics)
    b = np.argmax(a)
    c = matutils.sparse2full(vec2, atm.num_topics)
    d = np.argmax(c)
    dist = matutils.hellinger(matutils.sparse2full(vec1, atm.num_topics), matutils.sparse2full(vec2, atm.num_topics))
    sim = 1.0 / (1.0 + dist)
    return sim

```

Kode 5.36 Hellinger Distance *LDA*

Untuk menghitung nilai similaritas vektor probabilitas menggunakan *Cosine Similarity* dilakukan dengan *library genism.matutils cossim.* Implementasi *Cosine similarity* sesuai dengan Kode 5.37

```
from gensim import matutils
from sklearn.metrics.pairwise import cosine_similarity
from scipy import spatial

author_vecs = [atm.get_author_topics(author) for author in atm.id2author.values()]
```

```
6.
7. # Get similarity between two vectors
8. def similarity(vec1, vec2):
9.     a = matutils.sparse2full(vec1, atm.n
   um_topics)
10.     b = np.argmax(a)
11.     c = matutils.sparse2full(vec2, atm.n
   um_topics)
12.     d = np.argmax(c)
13.     dist = matutils.cossim(vec1,vec2)
14.     sim = 1.0 / (1.0 + dist)
15. return sim
16.
```

Kode 5.37 Cossine Similarity LDA

Untuk menghitung nilai similaritas vektor probabilitas menggunakan *topic similarity* dilakukan dengan menggunakan *library difflib*. Sebelum dilakukan penghitungan, terlebih dahulu vektor probabilitas tersebut diubah menjadi bentuk array, lalu dicari nilai vektor probabilitas dari topik yang paling dominan dari dokumen dan *author*. Setelah didapat nilai vektor dari topik yang paling dominan, maka dilakukan perhitungan similaritas dari kedua nilai vektor tersebut. Penghitungan nilai similaritas menggunakan *topic similarity* sesuai dengan Kode 5.38

```
1. from gensim import matutils
2. from sklearn.metrics.pairwise import cos
   ine_similarity
3. from scipy import spatial
4.
5. author_vecs = [atm.get_author_topics(aut
   hor) for author in atm.id2author.values(
   )]
6.
7. # Get similarity between two vectors
8. def similarity(vec1, vec2):
9.     a = matutils.sparse2full(vec1, atm.n
   um_topics)
10.     b = np.argmax(a)
```

```
    c = matutils.sparse2full(vec2, atm.num_topics)
    d = np.argmax(c)
    dist = SequenceMatcher(None, str(a[b]), str(c[d])).ratio()
    sim = 1.0 / (1.0 + dist)
    return sim

```

Kode 5.38 Topic Similarity

Untuk mengambil nilai vektor dari seluruh author dilakukan sesuai dengan Kode 5.39

```
# get similarity of vector to all authors
def get_sims(vec):
    sims = [similarity(vec, vec2) for vec2 in author_vecs]
    return sims

```

Kode 5.39 Memanggil Vektor *Author*

Selanjutnya dari hasil nilai similaritas untuk setiap metode akan ditampilkan dengan tabel yang berisikan *id author*, nama *author*, nilai kemiripan, dan jumlah dokumen yang dimiliki oleh *author* dalam model. Hasil tersebut akan diurutkan dari nilai skor similaritas tertinggi. Tahapan pemuatan hasil similaritas kedalam tabel dilakukan dengan menggunakan *library pandas* dan sesuai dengan Kode 5.40

```
import pandas as pd

# Get table with similarities, author names, and size
def get_table(i, top_n=10, smallest_author=1):
    # Get similarities.
    sims = []
    sims = get_sims(doc_vecs[i])

```

```
# Arrange author names, similarities, and author sizes in a list of tuples
    table = []
    for elem in enumerate(sims):
        author_name = atm.id2author[elem[0]]
        sim = elem[1]
        author_size=len(atm.author2doc[author_name])
        if author_size >= smallest_author:
            table.append((author_name, sim, author_size))

    # Make dataframe and retrieve top authors.
    df = pd.DataFrame(table, columns=['Author', 'Score', 'Size'])
    df = df.sort_values('Score', ascending=False)[:top_n]

    return df

# Get similarities between document and authors
for n in range(20):
    table = get_table(n)
    pprint('Model Stemming 50 Topik: Dokumen ' + str(n+1))
    print(table)
```

Kode 5.40 Visualisasi ke Dalam Tabel

5.9 Pengujian Model *Author-Topic Gaussian LDA* Dengan Menggunakan Nilai Similaritas Vektor Probabilitas

Pengujian model pada *author-topic gaussian LDA* dilakukan sesuai dengan yang dilakukan pada pengujian model pada *author-topic LDA*. Dimana tahapa ini dilakukan untuk menguji kemampuan prediksi dari model yang telah disimpan sebelumnya dalam mengetahui *author* yang membuat dokumen

uji. Hal ini dilakukan dengan menghitung nilai similarity dari vektor probabilitas yang dimiliki *author* dengan dokumen menggunakan tiga metode, yaitu metode *Hellinger distance, cosine similarity*, dan *topic similarity.*

5.9.1 Memuat Data Pengujian

Data pengujian yang digunakan pada tahap ini sama dengan data pengujian yang dilakukan pada tahap sebelumnya. Hal tersebut juga berlaku pada saat melakukan pembersihan data. Tahapan untuk memuat dan membersihkan data pengujian sesuai dengan Kode 5.33 dan Kode 5.34

5.9.2 Menghitung Probabilitas Kata dalam setiap topik pada dokumen uji

Setelah data selesai dibersihkan, selanjutnya dilakukan penghitungan nilai probabilitas distribusi kata pada setiap topik. nilai dari probabilitas kata pada topik tersebut akan dijumlahkan dan dibagi dengan jumlah kata yang ada pada topik tersebut, hal tersebut berlaku juga untuk topik lainnya. Setelah proses tersebut selesai dilakukan, maka disimpan kedalam variabel '*dokumenuji_topik'* sebagai vektor probabilitas dari dokumen uji. Tahapan ini dilakukan sesuai dengan Kode 5.41

```
dokumenuji_topik = []
for a in docs_test:
 topikuji = []
 filtered_doc = []
 nkd = defaultdict(float)
 for word in a.split():
  try:
   wvmodel[word]
   filtered_doc.append(word)  # Remove w
ords from doc that are not in word-
vec model
   nkd[lda.word_topics[word]] += 1.
  except KeyError:
   continue
 print(filtered_doc)
 print ("{} words removed from doc".form
at(len(filtered_doc) - len(a.split())))
```

```
word_topics = []
c = Counter(lda.word_topics.values())
docuji_topik = []
docuji_topics = []
for word in filtered_doc:
 posterior = []
 for k in range(lda.numtopics):
  prob = lda.draw_new_wt_assgns(word, k, wvmodel=wvmodel, new_doc=True) + log(lda.alpha + c[k])
  posterior.append(prob)
 posterior /= np.sum(posterior)
 docuji_topics.append(posterior)
for t in zip(*(docuji_topics)):
 docuji_topik.append((np.sum(t)))
docuji_topik /= np.sum(docuji_topik)
topikuji.append(docuji_topik)
del(docuji_topics)
dokumenuji_topik.append(docuji_topik)

```

Kode 5.41 Probabilitas Kata *GLDA*

5.9.3 Menghitung nilai similaritas vektor probabilitas

Pada tahapan ini juga dilakukan penghitungan nilai similaritas vektor probabilitas dari *author* dan dokumen dengan menggunakan tiga metode, yaitu *Hellinger distance*, *cosine similarity*, dan *topic similarity*. Untuk menghitung nilai similaritas vektor probabilitas menggunakan *Hellinger distance* dilakukan dengan *library genism.matutils Hellinger*. Implementasi *Helinger distance* sesuai dengan Kode 5.42

```
for n in range(20):
    hasil = []
    for lines in lda.author_top:
        vec1 = dokumenuji_topik[n]
        vec2 = lines[1]
        vec = gensim.matutils.full2sparse(vec1)
        vecs = gensim.matutils.full2sparse(vec2)
```

```
8.          dot = np.dot(vec1,vec2)
9.          norma = np.linalg.norm(vec1)
10.         normb = np.linalg.norm(vec2)
11.         cos = dot / (norma * normb)
12.
```

Kode 5.42 *Hellinger Distance GLDA*

Untuk menghitung nilai similaritas vektor probabilitas menggunakan *Cosine Similarity* dilakukan dengan *library genism.matutils cossim.* Sebelum dilakukan penghitungan nilai similaritas, terlebih dahulu nilai vektor probabilitas dari *author* dan dokumen diubah menjadi bentuk *bag of word* dengan menggunakan *library genism.full2sparse*. Implementasi *Cosine similarity* sesuai dengan Kode 5.43

```
1. for n in range(20):
2.     hasil = []
3.     for lines in lda.author_top:
4.         vec1 = dokumenuji_topik[n]
5.         vec2 = lines[1]
6.         vec = gensim.matutils.full2spars
   e(vec1)
7.         vecs = gensim.matutils.full2spar
   se(vec2)
8.         dot = np.dot(vec1,vec2)
9.         norma = np.linalg.norm(vec1)
10.        normb = np.linalg.norm(vec2)
11.        cos = matutils.hellinger(vec1,vec2)
12.
```

Kode 5.43 Cosine Similarity

Untuk menghitung nilai similaritas vektor probabilitas menggunakan *topic similarity* dilakukan dengan menggunakan *library difflib.* Sebelum dilakukan penghitungan, terlebih dahulu dicari nilai vektor probabilitas dari topik yang paling dominan dari dokumen dan *author*. Setelah didapat nilai vektor dari topik yang paling dominan, maka dilakukan perhitungan similaritas dari kedua nilai vektor tersebut. Penghitungan nilai

similaritas menggunakan *topic similarity* sesuai dengan **Kode 5.21**

```
for n in range(20):
    hasil = []
    for lines in lda.author_top:
        vec1 = dokumenuji_topik[n]
        vec2 = lines[1]
        vec = gensim.matutils.full2sparse(vec1)
        vecs = gensim.matutils.full2sparse(vec2)
        dot = np.dot(vec1,vec2)
        norma = np.linalg.norm(vec1)
        normb = np.linalg.norm(vec2)
        cos = SequenceMatcher(None, str(vec1[np.argmax(vec1)]), str(vec2[np.argmax(vec2)])).ratio()

```

Kode 5.44 Topic Similarity

Selanjutnya dari hasil nilai similaritas untuk setiap metode akan ditampilkan dengan tabel yang berisikan *id author*, nama *author*, nilai kemiripan, dan jumlah dokumen yang dimiliki oleh *author* dalam model. Hasil tersebut akan diurutkan dari nilai skor similaritas tertinggi. Tahapan pemuatan hasil similaritas kedalam tabel dilakukan dengan menggunakan *library pandas* dan sesuai dengan Kode 5.1

```
import pandas as pd
from gensim import matutils
from difflib import SequenceMatcher
for n in range(20):
    hasil = []
    for lines in lda.author_top:
        vec1 = dokumenuji_topik[n]
        vec2 = lines[1]
        vec = gensim.matutils.full2sparse(vec1)
        vecs = gensim.matutils.full2sparse(vec2)
```

```
11.         dot = np.dot(vec1,vec2)
12.         norma = np.linalg.norm(vec1)
13.         normb = np.linalg.norm(vec2)
14.         #cos = dot / (norma * normb)
15.         #cos = matutils.hellinger(vec1,v
    ec2)
16.         cos = SequenceMatcher(None, str(
    vec1[np.argmax(vec1)]), str(vec2[np.argm
    ax(vec2)])).ratio()
17.         #cos = matutils.cossim(vec,vecs)

18.         author_name = lines[0]
19.         document = 'doc' + str(n+1)
20.         author_size = len(lda.author2doc
    [lines[0]])
21.         if author_size > 0:
22.             hasil.append((author_name,co
    s,author_size,document))
23.
24. df = pd.DataFrame(hasil, columns = ['Aut
    hor','vector','author_size','documentn']
    )
25.     df = df.sort_values('vector', ascend
    ing=True)[:10]
26.     #print(dp)
27.     print(df)

28.
```

Kode 5.45 Visualisasi ke dalam Tabel *GLDA*

BAB VI
HASIL DAN PEMBAHASAN

Bab ini menjelaskan hasil dari pengolahan data, pemodelan *author-topic* menggunakan *Gaussan LDA*, pemodelan *author-topic* menggunakan *LDA*, validasi kedua model, pembahasan hasil tahap pengujian kedua model.

6.1 Mempersiapkan Data

Data publikasi yang diunduh dari repository *ACL Anthology* dengan jarak waktu tahun 2014 hingga tahun 2018 adalah sebanyak 1003 dokumen untuk dokumen pelatihan dan 20 dokumen untuk dokumen uji. Jumlah dokumen, kata, dan *author* yang digunakan untuk pelatihan sesuai dengan Tabel **6.11.**

Tabel 6.1 Jumlah Data Asli

Jumlah Dokumen	**Jumlah Kata**	**Jumlah *Author***
1003	**5663375**	**2266**

6.2 Pra-Proses Data

Tahapan ini dilakukan pada data pelatihan. Tahap yang dilakukan adalah *data cleaning, case folding*, penghapusan *stopwords*, penghapusan kata dengan kode ascii lebih dari 128, dan penghapusan kata yang tidak terdapat dalam *word2vec* model. Perubahan jumlah kata pada saat sebelum dan sesudah dilakukan pra-proses data dapat dilihat pada Tabel 6.2

Tabel 6.2 Jumlah Data Pra-Proses

Tahapan	Jumlah Kata
Jumlah Asli	**5663375**
Cleaning Data	3325450
Penghapusan *Stopwords*	2263244
Penghapusan Ascii > 128	2262623

Penghapusan kata diluar *word2vec* model	2178951

6.3 Pembuatan *Dictionary* dari Dokumen

Pada tahap ini seluruh kata diambil dan dimasukkan kedalam *dictionary* dengan keluaran jenis kata unik dan *id* kata tersebut. Setelah dibentuk *dictionary* terdapat perbandingan jumlah unik kata dengan jumlah seluruh kata yang terlihat pada Tabel 6.3

Tabel 6.3 Jumlah Kata Unik

Tahapan	Seluruh Kata	Kata Unik
Jumlah Kata	2178951	28842

Jumlah kata unik ini yang digunakan sebagai *dictionary* pada proses pemodelan *author-topic* baik menggunakan *LDA* maupun *Gaussian LDA.*

6.4 Pemodelan *Author-Topic* Menggunakan Gaussian LDA

Pada tahap ini dilakukan analisis hasil dari pemodelan *author-topic* untuk mencari model terbaik berdasarkan nilai *perplexity* yang dilakukan berdasarkan eksperimen jumlah iterasi dan jumlah topik.

6.4.1 Pembentukan Vektor Kata

Pada tahap ini dilakukan tahapan mengubah kata menjadi vektor kata. Kata-kata yang terdapat pada *word2vec* model akan dihitung sebagai *useable_vocab.* Sedangkan kata yang tidak terdapat pada *word2vec* model akan dihitung sebagai *unuseable_vocab*. Jumlah kata yang dapat diubah menjadi vektor kata dapat dilihat pada Tabel 6.4

Tabel 6.4 Jumlah Vektor Kata

Jenis Kata	*Useable_vocab*	*Unuseable_vocab*
Jumlah Kata	28842	0

Jumlah data yang tidak dapat diubah menjadi vektor kata tidak ada, dikarenakan pada saat pra-proses data dilakukan penghapusan kata yang tidak tedapat dalam *word2vec*. Hal tersebut dilakukan untuk menyamakan jenis kata untuk

pemodel *author-topic* baik menggunakan *LDA* maupun *Gaussian LDA.*

6.4.2 Penentuan Jumlah Iterasi

Penentuan jumlah iterasi dilakukan berdasarkan analissi terhadap kestabilan nilai perplexity setiap iterasi. Percobaan yang dilakukan dengan menggunakan nilai parameter 10 iterasi. Jumlah topik yang digunakan adalah 225 topik dan 125 topik. Hasil eksperimen kemudian dicatat dan ditampilkan dalam bentuk Diagram 6.1

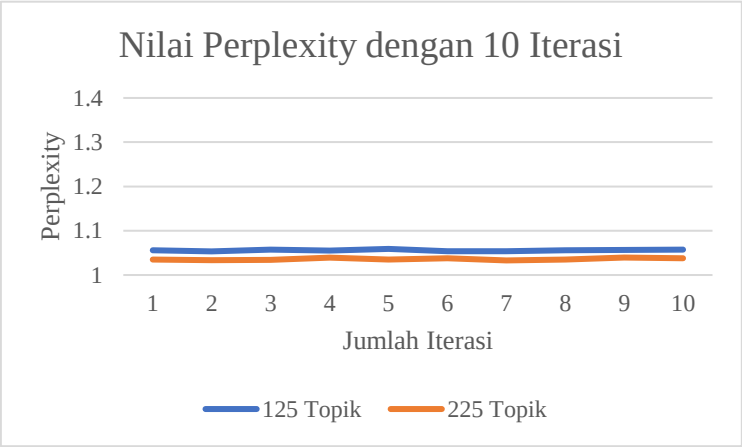


Diagram 6.1 ***Nilai Perplexity Iterasi GLDA***

Dalam 10 iterasi yang dilakukan, nilai perplexity untuk pemodelan *author-topic* menggunakan *Gaussian LDA* sudah terlihat stabil dari awal. Dari hasil tersebut, diputuskan nilai parameter iterasi yang digunakan adalah 2 iterasi.

6.4.3 Penentuan Jumlah Topik

Setelah didapatkan nilai parameter iterasi, selanjutnya eksperimen jumlah topik. percobaan dilakukan dengan menggunakan nilai parameter 2 iterasi. Sedangkan untuk eksperimen jumlah topik yang digunakan adalah 25, 50, 75, 100, 125, 150, 175, 200, 225, 250. Hasil eksperimen kemudian dicatat dan ditampilkan dalam bentuk Diagram 6.2

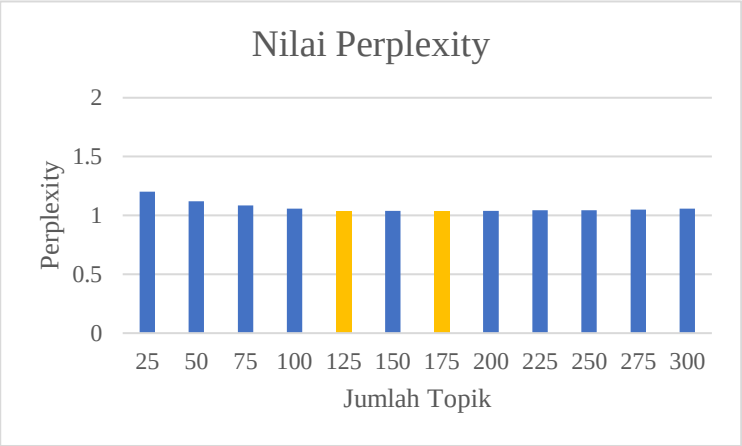


Diagram 6.2 Nilai Perplexity Topik GLDA

Dalam Diagram 6.2 dapat dilihat bahwa topik dengan nilai parameter 125 dan 175 memiliki nilai *perplexity* yang terkecil. Dengan hasil tersebut, maka penggunaan model untuk pengujian menggunakan parameter jumlah topik sebesar 125 dan 175 topik.

6.5 Pemodelan *Author-Topic* Menggunakan LDA

Pada tahap ini dilakukan analisis hasil dari pemodelan *author-topic* untuk mencari model terbaik berdasarkan nilai *perplexity* yang dilakukan berdasarkan eksperimen jumlah iterasi dan jumlah topik.

6.5.1 Penentuan Jumah Passes

Penentuan jumlah iterasi dilakukan berdasarkan analisis terhadap nilai perplexity setiap iterasi. Nilai parameter iterasi yang digunakan nantinya adalah iterasi yang nilai *perplexity* sudah terlihat stabil. Percobaan ini dilakukan dengan menggunakan nilai parameter *passes* awal 100. Jumlah topik yang digunakan adalah 25, 50, 75, 100, 125, 150, 175, 200, 225. Hasil nilai perplexity dari tiap passes dicatat dan ditampilkan dalam Diagram 6.3

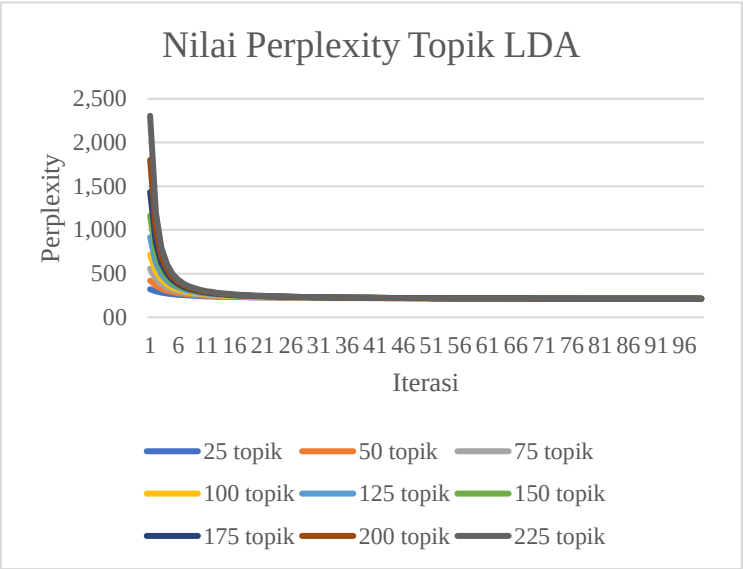


Diagram 6.3 Nilai Perplexity Iterasi LDA

Terlihat pada Diagram 6.3, bahwa nilai perplexity turun drastic pada iterasi awal. Sedangkan nilai perplexity untuk seluruh topik akan stabil pada pertengahan hingga akhir iterasi. Dengan hasil tersebut maka dapat disimpulkan untuk menggunakan nilai parameter iterasi sebesar 75. Hasil eksperimen penentuan jumlah passes dtampilkan pada Lampiran A.

6.5.2 Penentuan Jumlah Topik

Pada penentuan jumlah topik, juga dilakukan analisis terhadap nilai perplexity. Namun, berbeda dengan analisis nilai perplexity yang dilakukan pada saat penentuan jumlah iterasi. Pada penentuan jumlah topik nilai perplexity dicatat nilai perplexity terakhir. Lalu pada eksperimen ini juga dilakukan sebanyak 10 kali training model dan dicatat nilai perplexity untuk setiap kali training. Seluruh nilai perplexity yang dicatat tersebut dicari rata-ratanya. Pada eksperimen ini dilakukan dengan menggunakan nilai iterasi 75 dan jumlah topik sebesar 25, 50 , 75, 100, 125, 150, 175, 200, 225. Hasil nilai perplexity untuk masing-masing topik dapat dilihat pada Diagram 6.4

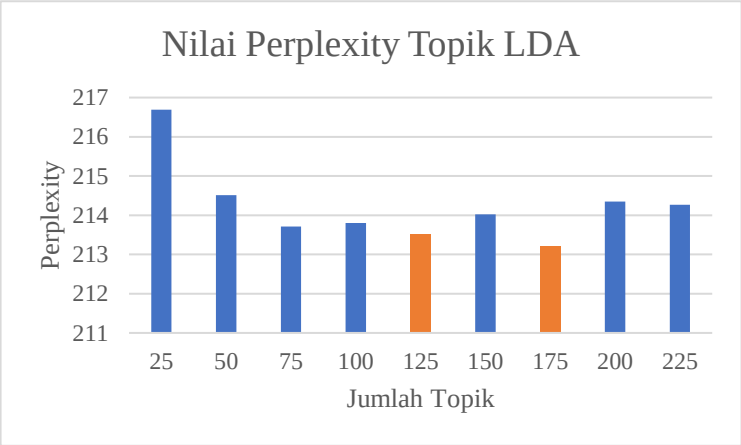


Diagram 6.4 Nilai Perplexity Topik LDA

Dari hasil Diagram 6.4 terlihat bahwa topik 125, dan 175 memiliki nilai perplexity yang baik. Selanjutnya dihitung nilai standard deviasi dari nilai perplexity tiap topik yang dapat dilihat pada Diagram 6.5

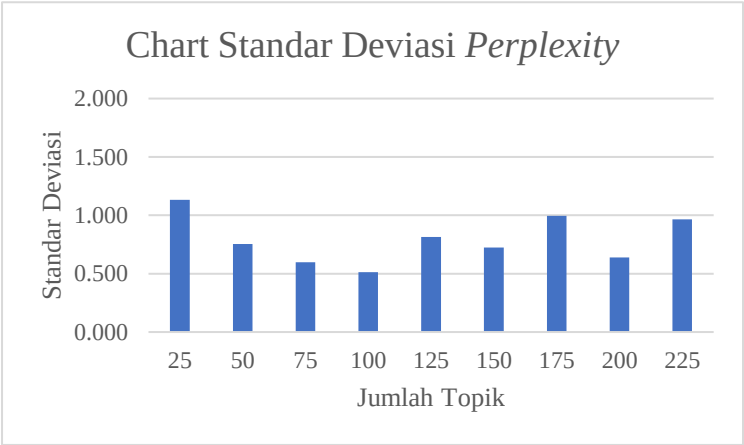


Diagram 6.5 Standar Deviasi Perplexity LDA

Berdasarkan Diagram 6.5 dapat dilihat standard deviasi dari beberapa topik. Hasil dari topik 125 dan 175 yang memiliki nilai *perplexity* terbaik sebelumnya memiliki nilai standar deviasi sebesar 0,8 dan 0,9. Jika dibandingkan dengan standard deviasi rata-rata seluruh topik sebesar 0,79, nilai standard deviasi topik tidak terlalu jauh.

6.6 Validasi Model *Author-Topic GLDA*

Pada validasi model dilakukan perhitungan nilai *topic coherence* dari setiap model mulai dari model dengan jumlah topik 125 dam 175. Tujuan menghitung seluruh nilai *topic coherence* model adalah untuk mengetahui pengaruh dari penggunaan *word embedding* terhadap data general dengan data spesifik untuk *author-topic model*.

Perhitungan nilai *topic coherence* dilakukan dengan menggunakan metode *pointwise mutual information* pada setiap topik yang ada di model. Seluruh nilai *topic coherence* yang dimiliki tiap topik dihitung rata-rata agar mendapatkan nilai *topik coherence* dari model. Nilai rata-rata *topic coherence* tiap model dapat dilihat pada Diagram **6.5**

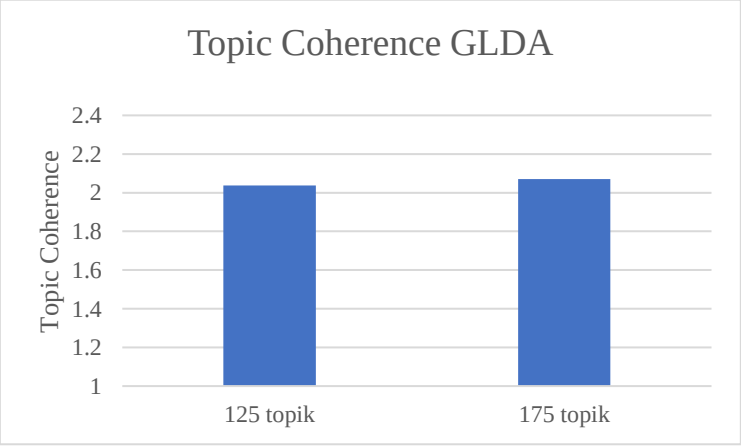


Diagram 6.6 Topic Coherence GLDA

Pada Diagram 6.6 dapat dilihat bahwa nilai *topic coherence* untuk kedua model terlihat stabil denngan nilai *topic coherence* masinng-masing adalah 2,037 dan 2,070. Nilai standar deviasi nilai *topic coherence* dapat dilihat pada Diagram 6.7

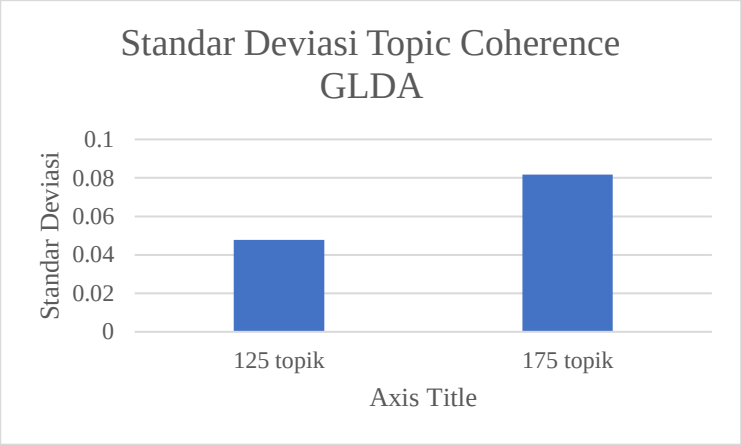


Diagram 6.7 Standar Deviasi Topic Coherence GLDA

Pada Diagram 6.7 diketahui bahwa standar deviasi untuk tiap model juga terlihat stabil, bahkan mendekati angka 0.

6.7 Validasi Model *Author-Topic LDA*

Pada tahap validasi ini dilakukan pencarian nilai *topic coherence* untuk jumlah topik 125 dan 175. Tahapan untuk mendapatkan nilai *topic coherence* pada model *LDA* adalah dengan memanggil *top-topics* dan menghitung nilai rata-rata *topic coherence* dari tiap topik. Rata-rata nilai *topic coherence* setiap model dapat dilihat pada Diagram 6.8

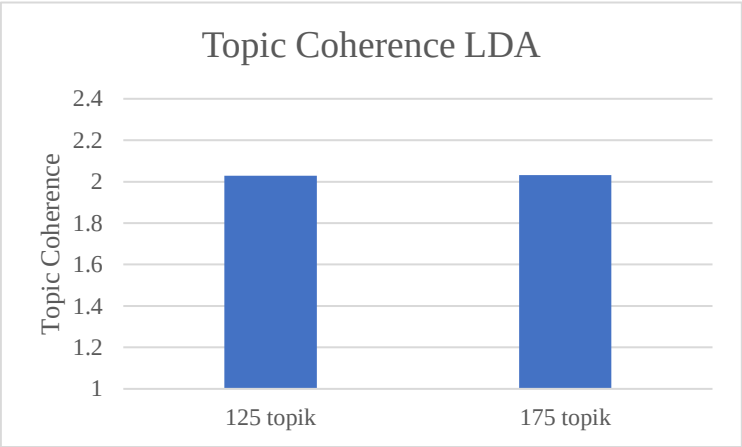


Diagram 6.8 Topic Coherence LDA

Dapat dilihat pada Diagram 6.8 bahwa nilai *topic coherence* pada tiap model terlihat stabil dengan nilai *topic coherence* masing-masing sebesar 2,027 dan 2,031. Nilai standar deviasi *topic coherence* dapat dilihat pada Diagram 6.9

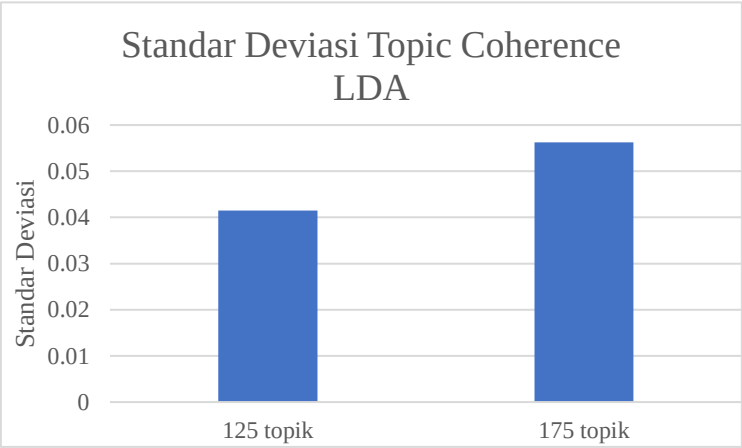


Diagram 6.9 Standar Deviasi Topic Coherence LDA

Pada Diagram 6.9, terlihat bahwa nilai standar deviasi untuk tiap model dengan jumlah topik yang berbeda sudah stabil. Hal

tersebut dapat menyimpulkan bahwa pesebaran kata pada tiap topik dapat dipahami dengan mudah oleh manusia.

6.8 Pengujian Model

Pada tahap ini dilakukan pengujian terhadap model yang telah dibuat sebelumnya baik dengan menggunakan *Gaussian LDA* maupun *LDA.* Data yang digunakan untuk pengujian sebanyak 220 dokumen yang belum digunakan sebelumnya pada pembuatan model. Syarat yang dibutuhkan dari dokumen untuk pengujian adalah *author* dari dokumen pengujian harus tercantum dalam model. Daftar data yang digunakan pada tahap pengujian model dapat dilihat pada Lampiran C

Pengujian ini dilakukan sebanyak 24 kali, dengan rincian dimana 2 kali dilakukan dengan menggunakan model *author-topic Gaussian LDA* dengan jumlah topik 125 dan 175 serta jumlah iterasi adalah 2. Pengujian lainnya dilakukan dengan menggunakan model *author-topic LDA* dengan jumlah topik 125 dan 175 serta jumlah iterasi adalah 75. Setiap model diuji sebanyak 4 kali menggunakan *Hellinger distance, Cosine similarity, topic similarity* dengan jumlah *author size* sebesar 1, 3, 5, 8. Minimal *author size* merupakan jumlah dokumen yang dimiliki oleh *author* di dalam model pelatihan. Contoh tabel hasil pengujian dokumen dengan metode *Hellinger distance* dengan minimal *author size* sebesar 1 dapat dilihat pada Gambar 6.1

'Model LDA 125 Topik: Dokumen 2'

	Author	Score	Size
1858	TomKenter	0.492827	1
703	HodLipson	0.492738	1
2246	ZhouYu	0.492384	1
1245	MatthewR.Gormley	0.492089	2
932	JonathanWintrode	0.490980	1
1426	OndrejDusek	0.489840	1
604	GiuseppeCarenini	0.489671	1
979	JunruShao	0.489241	1
930	JonathanR.Brennan	0.488717	1
2208	ZheZhang	0.488544	1

Gambar 6.1 Hasil Pengujian

Model yang diuji pada Gambar 6.1 merupakan model *author-topic* menggunakan *LDA* dengan jumlah topik sebesar 125 dan

minimal *author size* sebesar 1. Dokumen uji yang digunakan adalah dokumen 2 dengan judul "*Adaptive Recursive Neural Network for Target-dependent Twitter Sentiment Classification".* Penulis dari dokumen ini adalah Li Dong, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, dan Ke Xu. Pada hasil pengujian tidak terlihat nama *author* asli dari dokumen uji. Tabel hasil hanya menampilkan 10 *author* yang memiliki kemiripan paling dekat dengan dokumen uji.

Setelah dilakukan pengujian terhadap 10 *author* yang memiliki kemiripan dengan dokumen uji, hasil akurasi prediksi dari setiap model disimpan untuk dihitung nilai recall. Rumus perhitungan nilai *recall* sesuai dengan persamaan(1).

$$recall = \frac{jumlah\ dokumen\ yang\ diprediksi\ benar}{jumlah\ seluruh\ dokumen\ uji}\ x\ 100\%\ (1)$$

Tiap *author* dari dokumen uji yang terprediksi dengan benar akan dicatat. Lalu tiap dokumen yang terprediksi dengan benar akan dibagi dengan keseluruhan dokumen uji yaitu 220. Hasil perhitungan kemampuan prediksi model untuk setiap pengujian yang dilakukan pada kedua model dapat dilihat pada Tabel 6.5

Tabel 6.5 Hasil Prediksi

ATM - LDA						
min author size	HD	CS	MANUAL	HD	CS	MANUAL
	125			175		
1	1.36 %	0.91 %	0.00%	0.00 %	0.00 %	0.45%
3	0.00 %	0.00 %	9.09%	3.64 %	0.00 %	0.00%
5	5.45 %	5.00 %	0.00%	6.36 %	6.36 %	0.00%
8	5.45 %	5.45 %	3.64%	6.36 %	6.36 %	8.18%

ATM - GLDA						
min author size	HD	CS	MANUAL	HD	CS	MANUAL
	125			175		
1	2.27 %	0.00 %	0.00%	0.00 %	0.00 %	0.45%
3	4.55 %	4.55 %	9.09%	2.27 %	0.00 %	0.45%
5	6.82 %	6.36 %	0.45%	4.55 %	4.55 %	0.00%
8	7.73 %	5.91 %	3.64%	6.36 %	5.91 %	7.27%

Pada Tabel 6.5 perhitungan *similarity* dilakukan dengan menggunakan 3 metode, yaitu *Hellinger Distance (HD)*, *Cossine Similarity*, *Manual.* Perhitungan *HD* dilakukan dengan menghitung jarak seluruh *probability* yang dimiliki oleh dokumen uji dengan *author* pada model. Sedangkan perhitungan *CS* dilakukan dengan menghitung sudut kedekatan antara seluruh *probability* yang dimiliki dokumen uji dengan *author* pada model. Pada perhitungan *Manual* dilakukan dengan mencari indeks topik yang paling dominan dari dokumen uji, lalu dilakukan perhitungan antara nilai probability dengan indeks tersebut dari dokumen uji dengan *author* pada model.

Berdasarkan seluruh hasil pengujian yang telah dilakukan, dapat dilihat pada Lampiran C, bahwa jumlah minimal *author size* dapat memengaruhi tingkat keberhasilan model memberikan prediksi yang sesuai. Hal ini dapat dilihat dari hasil tiap *author size*, terlihat bahwa kedua model memiliki presentase yang tinggi pada nilai minimal *author size* diatas 7.

6.9 Analisis Berdasarkan Nilai *Perplexity*

Pada tahap ini akan dilakukan analisis perbandingan dari nilai *perplexity* untuk kedua metode dengan menggunakan parameter jumlah topik adalah 25, 50, 75, 100, 125, 150, 175, 200, 225. Terlihat bahwa pada Diagram 6.2 hasil rata-rata nilai *perplexity* dari seluruh parameter jumlah topik yang ada pada model *author-topic GLDA* sangat stabil atau bisa dikatakan bahwa persebaran kata pada setiap topik selalu merata. Sedangkan

pada Diagram 6.5 terlihat bahwa nilai *perplexity* dari model *author-topic LDA* terlihat tidak stabil atau dapat dikatakan persebaran kata pada topik bergantung pada parameter jumlah iterasi yang diberikan. Hal tersebut dapat disimpulkan bahwa penggunaan *word embedding* dapat memengaruhi distribusi kata pada topik.

6.10 Analisa Berdasarkan Nilai Topic Coherence

Pada tahap ini dilakukan analisa perbandingan nilai *topic coherence* untuk kedua model dengan menggunakan input parameter topik sebesar 125 dan 175. Setelah dilakukan perhitungan *topic coherence* menggunakan metode *pointwise mutual information*. Dapat dilihat pada Diagram 6.6 dan 6.8, bahwa dari kedua model memiliki nilai *topic coherence* yang dapat dikatakan berdekatan. Hal tersebut dapat penulis katakan bahwa penggunaan data yang bersifat spesifik dapat mempengaruhi pesebaran kata pada setiap topik baik untuk model *author-topic* menggunakan *Gaussian LDA* maupun *LDA*.

6.11 Analisis Hasil Pengujian Model

Pada analisis ini akan dilakukan perbandingan antara hasil kemampuan prediksi dari model *author-topic* dengan *Gaussian LDA* maupun *LDA*. Parameter utama yang dilihat pada analisis ini adalah *author size* atau jumlah minimal dokumen yang terhebung dengan author pada model.

Pada Tabel 6.5 dapat terlihat bahwa kedua model sulit untuk memrediksi *author* dengan benar. Pada model *author-topic* menggunakan *LDA* hanya ada 3 dari 6 kali pengujian dengan minimal *author size* 1 yang berhasil diprediksi dengan benar. Sedangkan pada model yang menggunakan *Gaussian LDA* hanya 2 dari 6 kali pengujian dengan minimal *author size* 1 yang berhasil diprediksi dengan benar. Dari hasil tersebut dapat dilihat bahwa dengan menggunakan nilai minimal *author size* 1, kedua model memiliki perbedaan yang sangat sedikit. Hal tersebut dapat disebabkan oleh penggunaan data yang kurang bersih. Data dikatakan kurang bersih karena masih terdapat

banyak kata yang tidak diperlukan atau kurang sesuai dengan penelitian pada dokumen tersebut, walaupun sudah dilakukan pembersihan data. contoh dari data kurang bersih dapat dilihat pada gambar

Method	Matrix Form	+Features	K	Span	Nuclearity	Relation
Prior work						
1. HILDA (Hernault *et al.*, 2010)				**83.0**	68.4	54.8
2. TSP 1-1 (Joty *et al.*, 2013)				82.47	68.43	55.73
3. TSP SW (Joty *et al.*, 2013)				82.74	68.40	55.71
Our work						
4. Basic features	$\mathbf{A} = \mathbf{0}$	Yes		79.43	67.98	52.96
5. Word embeddings	Concatenation	No	75	75.28	67.14	53.79
6. NMF	Concatenation	No	150	78.57	67.66	54.80
7. Bag-of-words	$\mathbf{A} = \mathbf{I}$	Yes		79.85	69.01	60.21
8. DPLP	Concatenation	No	60	80.91	69.39	58.96
9. DPLP	Difference	No	60	80.47	68.61	58.27
10. DPLP	Concatenation	Yes	60	82.08	**71.13**	**61.63**
11. DPLP	General	Yes	30	81.60	**70.95**	**61.75**
Human annotation				88.70	77.72	65.75

Gambar 6.2 Kata yang Tidak Diperlukan

implemented evaluation metrics together plp codes published method prior work hilda tsp tsp work basic features word embeddings nmf plp plp plp plp human annotation matrix form concatenation concatenation concatenation difference concatenation general yes yes yes yes span relation table parsing results different models test results tsp hilda reprinted prior work concatenation construction superior difference comparison lines inconclusive merits general form suggests using projection matrix model interrelationships substantially improve simpler concatenation construction may figure shows performance changes different latent dimensions value employ grid search development set identify optimal concatenation performance overly sensitive general form performance decreases large recall section construction nine times many parameters concatenation large values likely experimental results table presents rst parsing results plp alternative versions plp outperform prior relation includes relatively simple systems whose features simply projection word count vectors edu addition features table improves performance leading absolute improvement around relation prediction span plp performs slightly worse prior systems employ richer syntactic contextual might especially helpful span shown line results basic

Gambar 6.3 Data yang Masih Kotor

Terlihat pada Gambar 6.2 terdapat beberapa kata yang tidak diperlukan karena kurang sesuai dengan penelitian pada dokumen tersebut. Namun pada Gambar 6.3 terlihat bahwa data yang digunakan untuk pemodelan, masih terdapat kata-kata tersebut. Kurang bersihnya dalam tahap *pre-processing* membuat hasil pengujian model kurang bagus, sesuai dengan yang dijelaskan sebelumnya.

Selain kebersihan Data, jumlah author yang hanya memiliki 1 dokumen pada model terlalu banyak. Diagram jumlah dokumen yang dimiliki *author* pada model dapat dilihat pada Diagram 6.10

Diagram 6.10 Diagram Dokumen yang Dimiliki Author

Pada Diagram 6.10 dapat terlihat bahwa mayoritas *author* pada model memiliki dokumen sebanyak 1. Hal tersebut dapat meneyebabkan kemampuan prediksi model. Jika ingin mendapatkan nilai prediksi yang baik, maka nilai parameter minimal *author size* harus sesuai. Contohnya Tabel 6.5 untuk hasil prediksi dari tiap nilai minimal *author size* pada setiap model. Pada nilai minimal *author size* lebih dari 1, terlihat bahwa model *author-topic* dengan menggunakan *Gaussian LDA* dapat melakukan prediksi dengan lebih baik. Terbukti dari nilai 3 dan 5, *LDA* hanya mampu memrediksi 2 dan 4 dari 12 kali pengujian. Sedangkan, pada *GLDA* mampu memrediksi 5 dan 5 dari 12 kali pengujian. Dari hasil tersebut dapat diartikan bahwa penggunaan *word embedding* dapat menyebabkan nilai vector probabilitas yang dimiliki *author* dengan dokumen menjadi dekat. Terlihat bahwa *word embedding* akan mendapatkan hasil prediksi yang lebih baik ketika minimal *author size*nya diperbesar, atau dapat diartikan *author* semakin sedikit dan probabilitas vektor masing-masing *author* menjadi lebih bervariasi, walaupun sedikit.

Halaman ini sengaja dikosongkan

BAB VII
KESIMPULAN DAN SARAN

Pada bab ini dijelaskan kesimpulan dan saran yang diperoleh selama melakukan penelitian dalam tugas akhir ini.

7.1 Kesimpulan

Dari penelitian pada tugas akhir ini berikut kesimpulan yang didapatkan:

1. Berdasarkan hasil eksperimen nilai *perplexity* untuk kedua metode, didapatkan kedua model memiliki nilai yang baik pada jumlah topik 125 dan 175. Dengan hasil tersebut maka digunakan jumlah topik 125 dan 175 untuk kedua model. Sedangkan dalam pengukuran iterasi, didapatkan untuk model *author-topic LDA* menggunakan passes 75, dan untuk model *author-topic GLDA* menggunakan iterasi sebesar 2.
2. Perhitungan *topic coherence* menggunakan metode *pointwise mutual information* pada kedua model didapatkan bahwa untuk model dengan jumlah topik 125 dan 175 hasilnya lebih baik pada *author-topic GLDA*. Selisih nilai dari jumlah topik 125 adalah 0.01 dan pada jumlah topik 175 sebesar 0.039.
3. Data jurnal penelitian *ACL* memiliki banyak contoh dan hasil pada teks kata didalamnya. Hal tersebut membuat pesebaran kata yang ada pada topik kurang sesuai dengan penelitian yang ada pada data training. Pesebaran kata pada tiap topik dapat dilihat pada Lampiran B.
4. Pengujian prediksi *author* dari dokumen uji pada kedua model memiliki perbedaan yang sedikit. Hal tersebut dipengaruhi oleh penggunaan data yang terlalu spesifik terlihat dari jenis kata uniknya.

7.2 Saran

Berdasarkan penelitian yang telah dilakukan pada tugas akhir ini berikut merupakan saran dan masukan yang dapat digunakan sebagai acuan untuk penelitian berikutnya:

1. Jenis kata unik yang dimiliki untuk pelatihan dirasa masih sedikit dibandingkan dengan kata yang ada pada *word2vec*, sehingga perlu data yang lebih banyak lagi atau lebih bervariasi untuk melakukan *author-topic modelling* menggunakan *word embedding*
2. Penggunaan data dari penelitian hanya diambil isi dari penelitian tersebut saja. Teks seperti contoh dan isi tabel seharusnya dihapus.
3. Mencari metode *word embedding* lain untuk melakukan *author-topic modelling* sebagai pembanding metode *GLDA.*

DAFTAR PUSTAKA

[1] M. Rosen-Zvi, T. Griffiths, M. Steyvers and P. Smyth, "The Author-Topic Model for Authors and Documents," pp. 487-494, 2004.

[2] R. Das, M. Zaheer and C. Dyer, "Gaussian LDA for Topic Models with Word Embeddings," in *Proc. ACL 2015*, 2015.

[3] M. D. Blei and A. Y. J. M. I. Ng, "Latent Dirichlet Allocation," *Machine Learning Research*, pp. 993-1002, 2003.

[4] T. Mikolov, I. Sutskever, K. Chen, G. Corrado and J. Dean, "Distributed Representations of Words and Phrases and their Compositionality," 2013.

[5] M. Naili, A. H. Chaibi and H. Ghezala, "Comparative Study of Word Embedding Methods in Topic Segmentation," *Proc. Computer Science*, vol. 112, pp. 340-349, 2017.

[6] K. Dhruvil, "Introdiction to Word Embedding and Word2Vec," Towards Data Science, 1 September 2018. [Online]. Available: https://towardsdatascience.com/introduction-to-word-embedding-and-word2vec-652d0c2060fa. [Accessed 8 February 2019].

[7] W. Hu and J. Tsujii, "A Latent Concept topic Model for Robust Topic Inference Using Word Embeddings," *Proc of the 54th Annual Meeting of the Association for Computational Linguistic*, pp. 380-386, 2016.

[8] Anonymous, "A Bayesian Nonparametric Topic Model with Variational Auto-Encoders," in *Proc. ICLR* , New Orleans, 2018.

[9] R. Ding, R. Nallapati and B. Xiang, "Coherence-Aware Neural Topic Modelling," *Empirical Methods in Natural Language Processing*, pp. 830-836, 2018.

[10] K. Stevens, P. Kegelmeyer and D. B. D. Andrezejewski, "Exploring Topic Coherence Over Many Models and Many Topics," *Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pp. 952-962, 2012.

[11] M. Markatou, Y. Chen, G. Afendras and G. Lindsay, "Statistical Distance and Their Role in Robustness," *math*, pp. 1-23, 2016.

LAMPIRAN A – Eksperimen Penentuan Jumlah Topik Berdasarkan Nilai Perplexity

Tabel Hasil Eksperimen Pnentuan Jumlah *Passes Latent Dirichlet Allocation*

Iterasi	Jumlah Topik								
	25	50	76	100	125	150	175	200	225
1	4174.9	9019.8	21884.1	57499	160726	471348	1440800.8	4560106.9	14877846.2
2	319.8	419.9	555.8	714.1	916.1	1163.2	1433.5	1800.1	2305.5
3	298.3	361.6	439.4	522.7	621.8	737.5	854	1004.7	1197.8
4	283.6	327.5	379.3	431.2	490.6	557.4	619.6	700.3	798.8
5	272.7	304.7	342.0	377.1	416.5	459.2	496.7	545.8	604.1
6	264.4	288.6	317.0	342.2	370.3	399.8	424.3	456.9	494.9
7	258.0	276.7	299.3	318.3	339.4	361.1	377.9	401	427.6
8	252.8	267.8	286.4	301.1	317.6	334.3	346.4	363.4	383
9	248.6	260.8	276.5	288.2	301.6	314.9	323.8	336.9	351.9
10	245.2	255.2	268.7	278.3	289.4	300.3	307	317.4	329.2

11	242.3	250.7	262.5	270.5	279.8	289	294.1	302.5	312.1
12	239.9	247.0	257.4	264.1	272.1	280	283.9	290.8	298.8
13	237.8	243.9	253.2	258.9	265.9	272.7	275.6	281.5	288.2
14	236.0	241.2	249.6	254.5	260.6	266.7	268.9	273.9	279.6
15	234.4	238.9	246.5	250.7	256.2	261.6	263.3	267.6	272.5
16	233.0	236.9	243.8	247.5	252.4	257.3	258.5	262.3	266.5
17	231.8	235.2	241.5	244.8	249.2	253.6	254.4	257.7	261.5
18	230.7	233.6	239.4	242.3	246.3	250	250.9	253.9	257.2
19	229.7	232.2	237.6	240.2	243.8	247.6	247.9	250.5	253.4
20	228.8	231.0	235.9	238.2	241.6	245.1	245.2	247.5	250.2
21	228.0	229.9	234.4	236.5	239.7	242.9	242.8	244.9	247.3
22	227.3	228.8	233.1	235	237.9	241	240.7	242.6	244.8
23	226.6	227.9	231.9	233.6	236.3	239.2	238.8	240.5	242.5
24	225.9	227.1	230.8	232.3	234.9	237.6	237.1	238.6	240.5
25	225.4	226.3	229.8	231.2	233.5	236.1	235.5	237	238.7

26	224.3	225.5	228.8	230.1	232.3	234.8	234.1	235.4	237
27	223.9	224.9	228.0	229.1	231.2	233.6	232.8	234	235.5
28	223.4	224.3	227.2	228.2	230.2	232.5	231.7	232.8	234.1
29	223.0	223.7	226.4	227.4	229.3	231.4	230.6	231.6	232.9
30	222.6	223.1	225.8	226.6	228.4	230.5	229.6	230.5	231.7
31	222.3	222.6	225.1	225.9	227.6	229.6	228.6	229.5	230.7
32	221.9	222.1	224.5	225.3	226.9	228.8	227.8	228.6	229.7
33	221.6	221.7	224.0	224.6	226.2	228	227	227.8	228.8
34	221.3	221.3	223.4	224.1	225.5	227.3	226.2	227	227.9
35	221.0	220.9	222.9	223.5	224.9	226.6	225.5	226.2	227.1
36	220.7	220.5	222.5	223	224.3	226	224.9	225.6	226.4
37	220.5	220.1	222.0	222.5	223.8	225.4	224.3	224.9	225.7
38	220.2	219.8	221.6	222.1	223.3	224.8	223.7	224.3	225
39	220.0	219.5	221.2	221.6	222.8	224.3	223.2	223.7	224.4
40	219.7	219.2	220.8	221.2	222.3	223.8	222.6	223.2	223.9

41	219.5	218.9	220.5	220.8	221.9	223.4	222.2	222.7	223.3
42	219.3	218.6	220.2	220.5	221.5	222.9	221.7	222.2	222.8
43	219.1	218.4	219.8	220.1	221.1	222.5	221.3	221.7	222.3
44	218.9	218.1	219.5	219.8	220.7	222.1	220.9	221.3	221.8
45	218.7	217.9	219.2	219.4	220.3	221.7	220.5	220.9	221.4
46	218.5	217.6	218.9	219.1	220	221.3	220.1	220.5	221
47	218.4	217.4	218.7	218.9	219.7	221	219.7	220.1	220.6
48	218.2	217.2	218.4	218.6	219.4	220.6	219.4	219.7	220.2
49	218.0	217.0	218.2	218.3	219.1	220.3	219.1	219.4	219.9
50	217.9	216.8	217.9	218.1	218.8	220	218.8	219.1	219.5
51	217.7	216.6	217.7	217.8	218.5	219.7	218.5	218.8	219.2
52	217.6	216.4	217.5	217.6	218.3	219.4	218.2	218.5	218.9
53	217.5	216.3	217.3	217.3	218	219.2	217.9	218.2	218.6
54	217.3	216.1	217.1	217.1	217.8	218.9	217.6	217.9	218.3
55	217.2	215.9	216.9	216.9	217.5	218.6	217.4	217.6	218

56	217.1	215.8	216.7	216.7	217.3	218.4	217.1	217.4	217.7
57	217.0	215.6	216.5	216.5	217.1	218.2	216.9	217.1	217.5
58	216.8	215.5	216.3	216.3	216.9	217.9	216.7	216.9	217.2
59	216.7	215.3	216.1	216.1	216.7	217.7	216.5	216.7	217
60	216.6	215.2	216.0	216	216.5	217.5	216.3	216.4	216.8
61	216.5	215.0	215.8	215.8	216.3	217.3	216.1	216.2	216.5
62	216.4	214.9	215.6	215.6	216.1	217.1	215.9	216	216.3
63	216.3	214.8	215.5	215.5	215.9	216.9	215.7	215.8	216.1
64	216.2	214.7	215.3	215.3	215.8	216.7	215.5	215.6	215.9
65	216.1	214.6	215.2	215.2	215.6	216.6	215.3	215.4	215.7
66	216.0	214.4	215.1	215	215.4	216.4	215.1	215.3	215.5
67	215.9	214.3	214.9	214.9	215.3	216.2	215	215.1	215.3
68	215.8	214.2	214.8	214.7	215.1	216.1	214.8	214.9	215.2
69	215.8	214.1	214.7	214.6	215	215.9	214.7	214.7	215
70	215.7	214.0	214.5	214.5	214.8	215.7	214.5	214.6	214.8

71	215.6	213.9	214.4	214.3	214.7	215.6	214.4	214.4	214.7
72	215.5	213.8	214.3	214.2	214.6	215.5	214.2	214.3	214.5
73	215.4	213.7	214.2	214.1	214.4	215.3	214.1	214.1	214.4
74	215.4	213.6	214.1	214	214.3	215.2	213.9	214	214.2
75	215.3	213.5	214.0	213.9	214.2	215	213.8	213.8	214.1
76	215.2	213.4	213.8	213.7	214.1	214.9	213.7	213.7	213.9
77	215.2	213.3	213.7	213.6	213.9	214.8	213.5	213.6	213.8
78	215.1	213.3	213.6	213.5	213.8	214.7	213.4	213.4	213.6
79	215.0	213.2	213.5	213.4	213.7	214.5	213.3	213.3	213.5
80	215.0	213.1	213.4	213.3	213.6	214.4	213.2	213.2	213.4
81	214.9	213.0	213.3	213.2	213.5	214.3	213.1	213.1	213.3
82	214.8	212.9	213.3	213.1	213.4	214.2	213	213	213.1
83	214.8	212.9	213.2	213	213.3	214.1	212.8	212.8	213
84	214.7	212.8	213.1	212.9	213.2	214	212.7	212.7	212.9
85	214.7	212.7	213.0	212.8	213.1	213.9	212.6	212.6	212.8

86	214.6	212.6	212.9	212.8	213	213.8	212.5	212.5	212.7
87	214.5	212.6	212.8	212.7	212.9	213.7	212.4	212.4	212.6
88	214.5	212.5	212.7	212.6	212.8	213.6	212.3	212.3	212.5
89	214.4	212.4	212.7	212.5	212.7	213.5	212.2	212.2	212.4
90	214.4	212.4	212.6	212.4	212.6	213.4	212.1	212.1	212.3
91	214.3	212.3	212.5	212.3	212.5	213.3	212	212	212.2
92	214.3	212.2	212.4	212.3	212.4	213.2	212	211.9	212.1
93	214.2	212.2	212.4	212.2	212.4	213.1	211.9	211.8	212
94	214.2	212.1	212.3	212.1	212.3	213	211.8	211.7	211.9
95	214.1	212.0	212.2	212	212.2	212.9	211.7	211.7	211.8
96	214.1	212.0	212.1	212	212.1	212.9	211.6	211.6	211.7
97	214.0	211.9	212.1	211.9	212	212.8	211.5	211.5	211.6
98	214.0	211.9	212.0	211.8	212	212.7	211.4	211.4	211.5
99	214.0	211.8	211.9	211.8	211.9	212.6	211.4	211.3	211.5
100	213.9	211.8	211.9	211.7	211.8	212.5	211.3	211.2	211.4

Tabel Hasil Eksperimen Pnentuan Jumlah Topik *Latent Dirichlet Allocation*

Percobaan	Jumlah Topik								
	25	50	76	100	125	150	175	200	225
1	216.9	214.8	213.5	214.3	213.6	214.5	211.5	213.5	213.7
2	217	215.3	214.1	214.8	213.8	213.1	212	213.3	214
3	214.8	214	213.5	213.6	212.5	215	213.1	214.9	214.2
4	216	213.9	213.7	213.1	212.7	214.3	213.2	213.9	214.5
5	218.4	213.9	214.4	213.8	214	215.2	214.1	214.6	214.3
6	217.9	214	213.6	213.4	214.2	214	213.6	214.3	214.5
7	217.8	213.4	213.8	213.3	212.4	213.4	213.2	214.4	212
8	216.2	215.4	213.8	213.6	213	213.7	215.1	214.3	215.6
9	216.4	214.9	214.4	214	214.4	213.8	213.1	215.2	215
10	216.9	214.8	213.5	214.3	213.6	214.5	211.5	213.5	213.7
Rata-rata	216.69	214.51	213.71	213.8	213.52	214.02	213.2	214.35	214.27
Std Dev	1.133	0.755	0.597	0.512	0.813	0.724	0.997	0.640	0.964

LAMPIRAN B – 10 Kata Terbaik Pada Tiap Topik

Tabel 10 Kata Terbaik Pada Tiap Topik GLDA

TOPIC 0	TOPIC 1	TOPIC 2	TOPIC 3	TOPIC 4	TOPIC 5	TOPIC 6	TOPIC 7	TOPIC 8	TOPIC 9
perplexing	formula_1	only	baselines	performance	srl	chambless	must	smith	example
complexities	vector	on	unidirectional	success	pbi	diven	cannot	walker	similar
baffling	symmetric	one	bidirectional	achieved	cco	quilici	should	anderson	instance
puzzling	finite	ones	north-to-south	remarkable	csg	harbeson	able	moore	well
contradictions	formula_3	came	footway	best	tlm	murison	willing	robinson	as
trivial	formula_2	time	north/south	achievement	wfi	wickstrom	consider	allen	such
scenarios	formula_4	three	west-to-east	experience	andiamo	chanjindamanee	intend	miller	same
hypothetical	equivalently	first	west-east	impressive	igt	stratmann	agree	clark	used
fraught	formula_5	two	crosspiece	thanks	mtg	adeleke	fail	baker	different
vexing	formula_6	for	east-to-west	quality	bms	gosney	need	wilson	these

TOPIC 11	TOPIC 12	TOPIC 13	TOPIC 14	TOPIC 15	TOPIC 16	TOPIC 17	TOPIC 18	TOPIC 19	TOPIC 20
accuracy	enhance	frames	slinking	discourse	king	ultra-wideband	easiness	argumentative	dynamically
clarity	improve	cardboard	darting	normative	ii	wfi	unexpectedness	adversarial	mechanically
relevance	achieve	laminated	peeking	philosophical	reign	crsp	obtuseness	averse	analytically

mastery	ensuring	folding	boundin g	metaphysical	emperor	mct	imprecision	inherently	optimally
reliability	maintainin g	mesh	tiptoeing	metaphysics	iii	iccp	impenetrabilit y	overly	transparentl y
subjective	flexibility	stacked	zipping	aesthetics	iv	fst	oddness	humorless	externally
coherence	improving	plastic	careerin g	argumentatio n	empire	ldi	persuasivenes s	intellectually	haphazardly
comprehensio n	maintain	removabl e	careenin g	discursive	ruler	icat	defensiveness	timid	visualized
breadth	sustain	racks	slink	conceptions	throne	mcpa	topicality	pushy	efficiently
originality	ensure	molded	scooting	viewpoint	kingdo m	vpm	implausibility	judgmental	iteratively

TOPIC 21	TOPIC 22	TOPIC 23	TOPIC 24	TOPIC 25	TOPIC 26	TOPIC 27	TOPIC 28	TOPIC 29	TOPIC 30
i-l	data	max	step	tula	layer	dev	produce	que	loud
jhb	indicate	weber	to	camarines	layers	narayan	producing	ahora	shouting
ters	indicates	fritz	intended	luni	mixture	sharma	quantities	pero	crowd
ation	results	muller	necessary	gaya	add	prakash	amounts	donde	noisy
yyy	suggest	müller	provide	guayas	liquid	gopal	processing	sus	crowds
eur2004-fra	comparison	werner	meant	sorsogon	combine	mohan	extract	esta	listening
beb	indicating	schneider	take	vrbas	thick	gupta	extracted	fuera	angry
ōe-jī	estimate	jensen	make	selenge	coating	suri	materials	cuando	answering
eur2004-cze	analysis	lars	allow	tumbes	butter	kiran	extracting	por	cheering
mẹ	indicated	otto	possible	chamba	discard	verma	bulk	porque	loudly

Tabel 10 Kata Terbaik Pada Tiap Topik LDA

0	1	2	3	4	5	6	7	8	9
0.030*parsing	0.070*logical	0.036*dependency	0.031*players	0.021*learning	0.071*features	0.010*set	0.024*model	0.023*object	0.042*amr
0.019*pos	0.031*forms	0.027*parsing	0.030*image	0.017*state	0.029*feature	0.010*training	0.019*argument	0.018*image	0.037*concept
0.018*parser	0.016*form	0.019*features	0.029*games	0.016*action	0.028*user	0.008*system	0.016*alignment	0.014*objects	0.027*concepts
0.011*dependency	0.014*correct	0.015*dependencies	0.024*annotation	0.011*actions	0.017*prediction	0.008*data	0.014*case	0.013*visual	0.015*graph
0.010*tags	0.012*features	0.014*tree	0.023*game	0.009*language	0.013*whether	0.007*using	0.013*predicate	0.010*using	0.013*parsing
0.010*model	0.012*table	0.013*parser	0.019*video	0.008*figure	0.012*table	0.006*used	0.010*joint	0.010*human	0.011*semantic
0.009*features	0.011*beam	0.012*word	0.018*images	0.008*human	0.012*users	0.006*word	0.009*analysis	0.009*generated	0.011*span
0.008*system	0.011*semantic	0.012*feature	0.015*items	0.008*agent	0.012*number	0.006*use	0.009*use	0.009*figure	0.011*figure
0.008*table	0.011*rule	0.011*head	0.015*player	0.008*policy	0.012*frequency	0.006*approach	0.009*arguments	0.009*descriptions	0.011*parser

11	12	13	14	15	16	17	18	19	20
0.077*relevance	0.034*topic	0.025*entity	0.022*latent	0.014*change	0.056*translation	0.015*model	0.022*name	0.020*topic	0.016*topic
0.037*scope	0.031*cognates	0.018*model	0.016*algorithm	0.013*copula	0.031*bleu	0.014*training	0.015*english	0.017*words	0.009*features
0.032*document	0.030*responses	0.012*selection	0.012*text	0.012*words	0.031*encoder	0.012*set	0.011*two	0.014*label	0.008*one

0.032*category	0.030*response	0.011*use	0.007*number	0.011*rhetorical	0.029*source	0.011*mentions	0.009*language	0.013*word	0.007*words
0.026*explanation	0.027*art	0.010*user	0.007*optimization	0.010*semantic	0.023*layers	0.011*probability	0.008*based	0.011*sentence	0.007*feature
0.024*task	0.027*performance	0.010*set	0.007*matrix	0.009*word	0.023*convolutional	0.011*linear	0.008*translations	0.010*labels	0.007*topics
0.021*negation	0.026*state	0.009*query	0.007*embeddings	0.008*scientific	0.022*deep	0.010*templates	0.008*translation	0.010*information	0.006*system
0.021*relevant	0.025*section	0.009*factoid	0.006*tree	0.008*changes	0.022*attn	0.009*name	0.008*tagging	0.009*new	0.006*models
0.019*filtering	0.023*test	0.008*node	0.006*inference	0.008*functions	0.022*citation	0.009*mention	0.008*approach	0.009*candidate	0.006*model
0.017*crawling'	0.023*system'	0.007*structured'	0.006*results'	0.008*used'	0.019*unsupervised'	0.009*template'	0.007*visual'	0.008*first'	0.006*prompt'

21	22	23	24	25	26	27	28	29	30
0.039*dependency	0.015*features	0.107*knowledge	0.018*argument	0.011*tweets	0.036*tensor	0.024*systems	0.016*model	0.028*document	0.033*lexical
0.018*graph	0.009*dependency	0.058*entity	0.015*couplet	0.010*table	0.017*user	0.021*system	0.010*use	0.025*learning	0.028*language
0.017*nodes	0.009*two	0.040*facts	0.013*generation	0.010*model	0.015*model	0.017*stance	0.009*system	0.020*documents	0.022*categories
0.014*node	0.008*graph	0.038*entities	0.011*noise	0.010*using	0.015*discussion	0.012*models	0.008*time	0.018*topic	0.020*grammar
0.014*path	0.008*parsing	0.033*rewriting	0.011*bleu	0.008*models	0.012*turns	0.012*posts	0.008*using	0.015*ranking	0.020*english
0.013*transition	0.007*algorithm	0.020*base	0.010*using	0.007*two	0.011*conversations	0.012*slot	0.007*set	0.014*online	0.020*native

0.011*system	0.007*text	0.020*fact	0.010*arguments	0.007*first	0.011*wikipedia	0.012*post	0.007*encoder	0.010*model	0.019*dependencies
0.010*syntactic	0.006*one	0.018*subject	0.009*neural	0.007*used	0.011*metadata	0.011*using	0.007*best	0.010*graph	0.018*reading
0.009*set	0.006*syntactic	0.018*type	0.009*expert	0.007*problem	0.011*discussions	0.011*lecture	0.007*mention	0.009*algorithm	0.018*pos
0.008*paths'	0.006*set'	0.017*common'	0.008*evaluation'	0.007*dataset'	0.010*turn'	0.011*provenance'	0.006*models'	0.009*two'	0.016*portuguese'

LAMPIRAN C – Hasil Percobaan Prediksi Author

Tabel Percobaan Prediksi Author Size = 1

‘Model LDA 125 Topik: Dokumen 1’	‘Hellinger Distance Model GLDA 125 Topik: Dokumen 1’
Author Score Size	Author vector author_size
932 JonathanWintrode 0.504717 1	1788 JeremyCole 0.999966 1
1240 MatthewBurgess 0.504133 1	1818 AngelaFan 0.999961 1
161 AsadSayeed 0.502966 1	1794 YannDauphin 0.999961 1
849 JianqiangMa 0.502917 1	262 MinhLuanNguyen 0.999961 1
979 JunruShao 0.502313 1	337 IvorW.Tsang 0.999961 1
697 HiroshiIshikawa 0.502128 1	42 KianMingA.Chai 0.999961 1
1857 TodorMihaylov 0.501642 1	1234 ArtemSokolov 0.999960 2
1462 PeterClark 0.501591 1	1330 ZhisongZhang 0.999959 2
1427 OnkarArunPandit 0.501481 1	1168 LianhuiQin 0.999959 2
1809 TakuKudo 0.501285 1	
‘Model LDA 125 Topik: Dokumen 2’	‘Hellinger Distance Model GLDA 125 Topik: Dokumen 2’
Author Score Size	Author vector author_size
703 HodLipson 0.499439 1	1619 WeiSong 0.999970 1
932 JonathanWintrode 0.499333 1	1526 LizhenLiu 0.999970 1
1857 TodorMihaylov 0.496756 1	

849 JianqiangMa 0.496526 1 450 DimitraGkatzia 0.496392 1 979 JunruShao 0.495150 1 1883 UtpalGarain 0.494887 1 930 JonathanR.Brennan 0.494416 1 1427 OnkarArunPandit 0.494266 1 1858 TomKenter 0.493809 1	613 ChunliangLu 0.999965 1 1523 EvanZheranLiu 0.999962 1 1448 KelvinGuu 0.999962 1 1643 DongWang 0.999962 2 90 NanDuan 0.999962 3 1138 JoakimNivre 0.999962 1 1725 ArturoArgueta 0.999961 1 156 HongzhaoHuang 0.999960 2
‘Model LDA 125 Topik: Dokumen 3’ Author Score Size 703 HodLipson 0.496843 1 1427 OnkarArunPandit 0.491011 1 1260 MeiTu 0.490418 1 1858 TomKenter 0.490395 1 932 JonathanWintrode 0.488040 1 786 JamesHenderson 0.487633 1 1245 MatthewR.Gormley 0.487599 2 1885 VanessaWeiFeng 0.486895 1 1883 UtpalGarain 0.486558 1 161 AsadSayeed 0.486240 1	‘Hellinger Distance Model GLDA 125 Topik: Dokumen 3’ Author vector author_size 1705 TianchengZhao 0.999966 2 1420 MaxineEskenazi 0.999966 2 2181 Yuan-FangWang 0.999965 1 53 QimingChen 0.999965 1 376 MuyunYang 0.999965 1 76 LeiCui 0.999965 1 456 JuayuanChao 0.999961 1 1418 JacobEisentein 0.999961 1 1578 UmashanthiPavalanathan 0.999961 1 1424 JimFitzpatrick 0.999961 1

‘Model LDA 125 Topik: Dokumen 4’ Author Score Size 703 HodLipson 0.495570 1 1857 TodorMihaylov 0.495111 1 932 JonathanWintrode 0.493116 1 1240 MatthewBurgess 0.490427 1 444 DianeJ.Litman 0.489810 1 737 I.Beltagy 0.489452 1 1858 TomKenter 0.488959 1 109 AnetteFrank 0.488827 1 1094 LeiLi 0.488609 1 1427 OnkarArunPandit 0.488293 1	‘Hellinger Distance Model GLDA 125 Topik: Dokumen 4’ Author vector author_size 968 YoavGoldberg 0.999975 3 921 IdoDagan 0.999974 4 316 NoahA.Smith 0.999974 12 212 ZhoujunLi 0.999974 4 1078 YankaiLin 0.999974 3 1531 HanLu 0.999974 1 1529 ChiehLo 0.999974 1 551 XuanjingHuang 0.999973 9 1167 PengQian 0.999973 2 395 MeishanZhang 0.999973 3
‘Model LDA 125 Topik: Dokumen 5’ Author Score Size 781 JamesFan 0.525170 1 1094 LeiLi 0.524443 1 932 JonathanWintrode 0.522420 1 1741 SiddharthaBanerjee 0.521806 1 347 ClaudiaPinto 0.520150 1 203 BertHuang 0.519380 1	‘Hellinger Distance Model GLDA 125 Topik: Dokumen 5’ Author vector author_size 1407 YannN.Dauphin 0.999952 1 1436 JonasGehring 0.999952 1 1006 JimmyXiangjiHuang 0.999950 1 1299 ZhiwenXie 0.999950 1 1828 ZihaoFu 0.999948 1

563	FrancouisRousseau	0.518800	1	1646	JamesZ.Wang 0.999948 1
409	DavidHarwath	0.518593	1	1664	ZhaohuiWu 0.999948 1
1640	SandraM.Aluisio	0.518387	1	1517	YanranLi 0.999948 1
241	BrigitteGrau	0.518066	1	1513	JiaLi 0.999948 1
				1682	JianboYe 0.999948 1

Tabel Percobaan Prediksi Author Size = 3

'Model LDA 125 Topik: Dokumen 1' Author Score Size 1 AdamLopez 0.491068 3 211 SrinivasanIyer 0.490815 3 70 GholamrezaHaffari 0.490288 3 62 ErikCambria 0.488514 3 194 RuslanSalakhutdinov 0.487800 3 228 WeiXu 0.487527 4 107 JieZhou 0.487434 3 146 MarekRei 0.486789 3 159 MirellaLapata 0.486500 9 154 Ming-WeiChang 0.486033 3	'Hellinger Distance Model GLDA 125 Topik: Dokumen 1' Author vector author_size 17 HaiLeongChieu 0.999958 3 232 HaoPeng 0.999956 3 222 JuliaKreutzer 0.999956 3 89 MarkSteedman 0.999955 4 116 HaiZhao 0.999955 5 264 AmirZadeh 0.999952 4 16 DavidChiang 0.999952 3 192 SiWei 0.999952 4 124 GuodongZhou 0.999952 6 238 DanRoth 0.999951 5
'Model LDA 125 Topik: Dokumen 2' Author Score Size 194 RuslanSalakhutdinov 0.485755 3 70 GholamrezaHaffari 0.484901 3 1 AdamLopez 0.484366 3 211 SrinivasanIyer 0.483818 3 107 JieZhou 0.483172 3	'Hellinger Distance Model GLDA 125 Topik: Dokumen 2' Author vector author_size 26 NanDuan 0.999962 3 259 GuopingHu 0.999958 3 53 JiweiLi 0.999958 4 16 DavidChiang 0.999956 3

62 ErikCambria 0.482974 3 146 MarekRei 0.481639 3 27 BowenZhou 0.480267 4 52 DongchanKim 0.479712 3 72 GregDurrett 0.479340 3	44 MarkJohnson 0.999955 4 156 KevinKnight 0.999955 6 265 QingyuZhou 0.999953 3 150 WeiHe 0.999952 4 61 Chin-YewLin 0.999952 6 170 AndreF.T.Martins 0.999951 4
'Model LDA 125 Topik: Dokumen 3' Author Score Size 62 ErikCambria 0.475711 3 1 AdamLopez 0.474167 3 154 Ming-WeiChang 0.474118 3 194 RuslanSalakhutdinov 0.472446 3 211 SrinivasanIyer 0.471961 3 159 MirellaLapata 0.470671 9 70 GholamrezaHaffari 0.470468 3 52 DongchanKim 0.469545 3 116 JunXu 0.469158 3 236 WilliamW.Cohen 0.468688 5	'Hellinger Distance Model GLDA 125 Topik: Dokumen 3' Author vector author_size 45 ShafiqJoty 0.999960 6 268 SebastianRuder 0.999958 3 212 AlexanderFraser 0.999958 3 147 JiaweiHan 0.999954 3 237 DavidReitter 0.999954 3 177 XiaomanPan 0.999954 3 137 YuboChen 0.999953 4 58 ZhenghuaLi 0.999951 4 76 PanupongPasupat 0.999951 5 119 DongdongZhang 0.999951 3
'Model LDA 125 Topik: Dokumen 4' Author Score Size	'Hellinger Distance Model GLDA 125 Topik: Dokumen 4'

1 AdamLopez 0.478084 3 95 IvanTitov 0.475717 3 194 RuslanSalakhutdinov 0.475709 3 70 GholamrezaHaffari 0.475621 3 110 JonasKuhn 0.475160 3 154 Ming-WeiChang 0.474477 3 211 SrinivasanIyer 0.474286 3 217 TiejunZhao 0.472999 4 228 WeiXu 0.472708 4 107 JieZhou 0.472626 3	Author vector author_size 215 YoavGoldberg 0.999975 3 209 IdoDagan 0.999974 4 87 NoahA.Smith 0.999974 12 59 ZhoujunLi 0.999974 4 224 YankaiLin 0.999974 3 153 XuanjingHuang 0.999973 9 121 MeishanZhang 0.999973 3 46 DanKlein 0.999973 9 13 BaobaoChang 0.999973 8 182 EkaterinaShutova 0.999972 3
'Model LDA 125 Topik: Dokumen 5' Author Score Size 150 MatteoNegri 0.508860 3 236 WilliamW.Cohen 0.507247 5 274 ZhoujunLi 0.506830 4 110 JonasKuhn 0.505551 3 16 BarbaraPlank 0.505199 4 45 DanielPreotiuc-Pietro 0.504883 3 184 RanTian 0.504796 3 272 ZhitingHu 0.504646 3	'Hellinger Distance Model GLDA 125 Topik: Dokumen 5' Author vector author_size 233 AriRappoport 0.999925 5 272 ParthaTalukdar 0.999923 3 25 MarkDredze 0.999922 3 193 DavidWeiss 0.999921 3 203 AntoineBosselut 0.999920 3 198 WenpengYin 0.999916 3 23 JacobDevlin 0.999915 3

201	ShashankSrivastava	0.504190	3
52	DongchanKim	0.504069	3

254	RuiZhang	0.999912	3
161	QuocV.Le	0.999911	3
0	OmriAbend	0.999910	8

Tabel Percobaan Prediksi Author Size = 7

'Model LDA 125 Topik: Dokumen 1' Author Score Size 22 MirellaLapata 0.486500 9 28 PhilBlunsom 0.483994 7 35 WilliamYangWang 0.482763 8 12 JianfengGao 0.478199 8 30 RobertoNavigli 0.477976 7 38 XuanjingHuang 0.477284 9 19 MinZhang 0.477167 7 2 ChristopherD.Manning 0.475542 12 42 ZhengdongLu 0.475035 7 3 ClaireCardie 0.474483 7	'Hellinger Distance Model GLDA 125 Topik: Dokumen 1' Author vector author_size 31 MinZhang 0.999950 7 30 JunZhao 0.999950 14 35 QunLiu 0.999949 8 17 KangLiu 0.999949 13 10 MinlieHuang 0.999949 7 22 MohammadTaherPilehvar 0.999948 7 1 MarcoBaroni 0.999947 8 15 MuLi 0.999947 7 36 YangLiu 0.999947 13 33 MingZhou 0.999946 18
'Model LDA 125 Topik: Dokumen 2' Author Score Size 22 MirellaLapata 0.477390 9 35 WilliamYangWang 0.476559 8 12 JianfengGao 0.472814 8 30 RobertoNavigli 0.472624 7 19 MinZhang 0.472181 7	'Hellinger Distance Model GLDA 125 Topik: Dokumen 2' Author vector author_size 26 WilliamYangWang 0.999951 8 18 TingLiu 0.999950 12 10 MinlieHuang 0.999949 7 6 YejinChoi 0.999948 10

28 PhilBlunsom 0.471930 7 38 XuanjingHuang 0.471136 9 36 XiaojunWan 0.468837 9 2 ChristopherD.Manning 0.468328 12 42 ZhengdongLu 0.468026 7	9 JianfengGao 0.999946 8 30 JunZhao 0.999945 14 41 XiaojunWan 0.999945 9 17 KangLiu 0.999945 13 19 ZhiyuanLiu 0.999945 9 0 OmriAbend 0.999944 8
'Model LDA 125 Topik: Dokumen 3' Author Score Size 22 MirellaLapata 0.470671 9 28 PhilBlunsom 0.467521 7 35 WilliamYangWang 0.465193 8 30 RobertoNavigli 0.465134 7 14 JunZhao 0.464318 14 12 JianfengGao 0.463839 8 19 MinZhang 0.463037 7 27 PercyLiang 0.462769 16 42 ZhengdongLu 0.460524 7 8 GrahamNeubig 0.460192 10	'Hellinger Distance Model GLDA 125 Topik: Dokumen 3' Author vector author_size 45 ShafiqJoty 0.999960 6 268 SebastianRuder 0.999958 3 212 AlexanderFraser 0.999958 3 147 JiaweiHan 0.999954 3 237 DavidReitter 0.999954 3 177 XiaomanPan 0.999954 3 137 YuboChen 0.999953 4 58 ZhenghuaLi 0.999951 4 76 PanupongPasupat 0.999951 5 119 DongdongZhang 0.999951 3
'Model LDA 125 Topik: Dokumen 4' Author Score Size	'Hellinger Distance Model GLDA 125 Topik: Dokumen 4'

22 MirellaLapata 0.472020 9 14 JunZhao 0.470406 14 35 WilliamYangWang 0.469677 8 33 VincentNg 0.466805 8 12 JianfengGao 0.465630 8 27 PercyLiang 0.465309 16 2 ChristopherD.Manning 0.464509 12 30 RobertoNavigli 0.464422 7 32 TingLiu 0.463826 12 28 PhilBlunsom 0.463702 7	Author vector author_size 24 NoahA.Smith 0.999974 12 34 XuanjingHuang 0.999973 9 13 DanKlein 0.999973 9 5 BaobaoChang 0.999973 8 40 XipengQiu 0.999972 8 25 WeiweiSun 0.999971 7 28 EricP.Xing 0.999971 10 8 ChrisDyer 0.999971 13 7 ShayB.Cohen 0.999971 7 11 EduardHovy 0.999971 13
'Model LDA 125 Topik: Dokumen 5' Author Score Size 19 MinZhang 0.494921 7 31 ShayB.Cohen 0.494456 7 14 JunZhao 0.494390 14 9 HengJi 0.493826 12 8 GrahamNeubig 0.493588 10 27 PercyLiang 0.493191 16 35 WilliamYangWang 0.492970 8 26 OmriAbend 0.492200 8	'Hellinger Distance Model GLDA 125 Topik: Dokumen 5' Author vector author_size 0 OmriAbend 0.999910 8 6 YejinChoi 0.999909 10 26 WilliamYangWang 0.999900 8 29 LukeZettlemoyer 0.999898 11 9 JianfengGao 0.999897 8 10 MinlieHuang 0.999896 7 19 ZhiyuanLiu 0.999894 9

37 XipengQiu 0.492074 8	28 EricP.Xing 0.999893 10
30 RobertoNavigli 0.490989 7	40 XipengQiu 0.999892 8
	41 XiaojunWan 0.999892 9a

BIODATA PENULIS

Penulis bernama Muchammad Andhika Supriyanto, lahir di Tuban, Jawa Timur pada tanggal 18 Oktober 1997. Merupakan anak pertama dari dua bersaudara. Penulis telah menempuh pendidikan formal di SD Negeri 03 Tuban dan SD Negeri 01 Unggulan Tuban , SMP Negeri 1 Tuban, dan SMA Negeri 1 Tuban.

Pada tahun 2015, penulis melanjutkan studi ke jenjang pendidikan yang lebih tinggi di Institut Teknologi Sepuluh Nopember sebagai mahasiswa Departemen Sistem Informasi, Fakultas Teknologi Informasi dan Komunikasi. Penulis terdaftar sebagai mahasiswa dengan nomor induk (NRP) 05211540000085. Sebagai mahasiswa penulis aktif pada organisasi dan kepantiaan di tingkat kampus. Pengalaman organisasi penulis adalah sebagai pemain, staf, dan presidium Unit Kegiatan Mahasiswa Victory Sepuluh Nopember Marching Corps (VSNMC). Sebagai staf dan kepala divisi di Lembaga Minat Bakat (LMB) ITS. Untuk pengalaman kepanitiaan, penulis berkontribusi pada acara Departemen Sistem Informasi yaitu Information Systems Expo (ISE) sebagai staf bidan *competition test maker* dan staff ahli bidang administrasi.

Guna menyelesaikan studi dan mendapatkan gelar Sarjana Komputer (S.Kom), penulis mengambil topik penelitian *Topic modelling* studi kasus penilitan *Association for Computational Linguistics* pada Laboratorium Akuisisi Data dan Diseminasi Informasi (ADDI).

Untuk pertanyaan seputar penelitian ini, penulis dapat dihubungi melalui e-mail muchammadandhikas@gmail.com.

Halaman ini sengaja dikosongkan