BAB II

LANDASAN TEORI

2.1 Text Mining

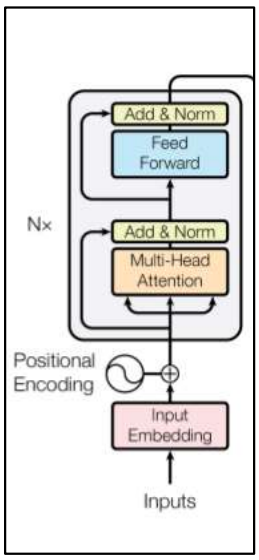
Text mining adalah proses menggali informasi di kumpulan teks yang banyak dan secara otomatis mengidentifikasi pola dan hubungan dengan data sebenarnya (Zhai dan Aggarwal, 2012). Data yang diolah merupakan *unstructured data* yang tidak dapat dipahami begitu saja oleh mesin. Dengan menggunakan *text mining*, data diolah sehingga dapat memberikan informasi yang dapat dipahami oleh mesin. Kemudian mesin dapat menganalisis informasi dan mengambil tindakan ataupun menghasilkan *output*. Klasifikasi teks berdasarkan topik merupakan salah satu contoh dari *output* yang dihasilkan mesin.

Proses *text mining* dimulai dengan mengumpulkan dokumen-dokumen yang akan dipakai (Gaikwad dkk., 2014). Dokumen-dokumen yang dikumpulkan akan diproses terlebih dahulu (*text preprocessing)* dengan mengecek format dan karakternya. Kemudian dokumen tersebut akan melalui *text analysis phase* untuk mendapatkan informasi dengan menggunakan teknik analisis teks. Beberapa contoh teknik analisis teks: *extraction, clustering, summarization.* Informasi yang didapat dapat disimpan di sistem manajemen informasi untuk memberikan *knowledge* kepada pengguna sistem.

Sebelum kumpulan teks dapat dipakai untuk menghasilkan informasi, harus dilakukan *text preprocessing*. Salah satu proses dalam *text preprocessing* adalah *tokenization* (Vijayarani dkk., 2015). *Tokenization* merupakan proses mengubah isi teks dokumen menjadi kumpulan kata individu.

2.2 ALBERT

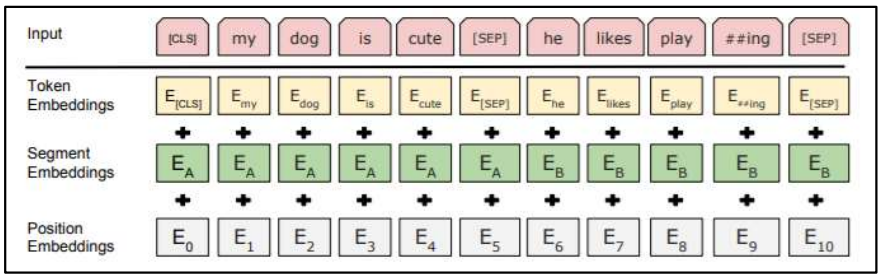
ALBERT (A Lite BERT) merupakan arsitektur *neural network* yang di-design dengan jumlah parameter lebih sedikit dibandingkan dengan arsitektur BERT (Lan dkk., 2019). Arsitektur ALBERT memiliki dasar yang sama dengan BERT yaitu menggunakan *transformer encoder* (Vaswani dkk., 2017) dan GELU *non linearities* (Hendrycks dan Gimpel, 2016). Gambar 2.1 menunjukkan arsitektur *transformer encoder.*



Gambar 2.1 Arsitektur Transformer Encoder (Vaswani dkk., 2017)

Transformer encoder memiliki dua sub-layer yaitu *multi-head self attention mechanism* dan *position-wise fully connected feed-forward network* (Vaswani dkk., 2017). *Multi-head self attention mechanism* melakukan kalkulasi *attention* lebih dari sekali sehingga memungkinkan model untuk memperhatikan informasi dari representasi vektor yang berbeda pada posisi yang berbeda. Pada *layer position-wise fully connected feed-forward network* dilakukan transformasi linear dengan menggunakan ReLU sebagai *activation function*. *Input embedding* yang dipakai pada *transformer encoder* adalah total penjumlahan *token embedding, segmentation*

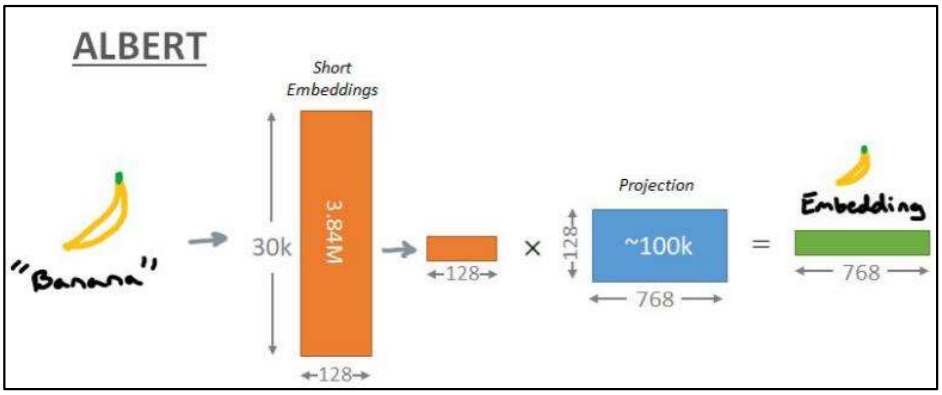
embedding, dan *position embedding* dari teks yang akan digunakan (Devlin dkk., 2018). Gambar 2.2 menunjukkan contoh visualisasi *input embedding*.



Gambar 2.2 *Input Embedding* (Devlin dkk., 2018)

Beberapa perubahan yang dilakukan pada arsitektur BERT adalah sebagai berikut (McCormick, 2020).

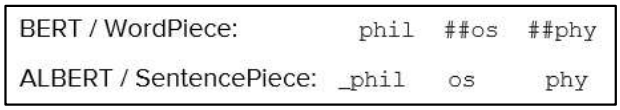
1. *Parameter Sharing,* pemakaian kembali *set of weights* pada seluruh *encoder layer* sehingga membuat ukuran ALBERT lebih kecil.
2. *Factorizing the Embeddings,* mengurangi jumlah *embedding* menjadi 128 fitur. Pada Gambar 2.3 menunjukkan visualisasi *embedding* kata pada ALBERT. Terdapat sebuah *projection matrix* di model ALBERT yang memperbesar ukuran *embedding* hingga 768 (McCormick, 2020).



Gambar 2.3 ALBERT *Embedding* (McCormick, 2020)

3. LAMB Optimizer untuk *pre-training*, menggantikan ADAM yang digunakan pada BERT.
4. Sentence Order Prediction (SOP), menggantikan Next Sentence Prediction yang digunakan pada BERT.
5. N-gram Masking, memodifikasi Masked Language Model (MLM) sehingga akan melakukan *masking* pada deretan *token*.

Selain itu, ALBERT menggunakan SentencePiece *tokenizer* berbeda dengan BERT yang menggunakan WordPiece *tokenizer* (McCormick, 2020). SentencePiece adalah sebuah *language-independent subword tokenizer* dan *detokenizer* (Kudo dan Richardson, 2018). Terdapat 4 komponen utama pada SentencePiece yaitu *Normalizer, Trainer, Encoder,* dan *Decoder*. *Normalizer* adalah sebuah modul untuk menormalisasi *semantically-equivalent* karakter *unicode* ke bentuk *canonical. Trainer* dipakai untuk melatih pembagian *subword* model dari *normalized corpus*. *Encoder* dipakai untuk membuat *token* dan *Decoder* digunakan untuk mengembalikan *token* menjadi teks semula. Perbedaan antara WordPiece dan SentencePiece dapat dilihat dari hasil tokenisasi untuk spasi yang memisahkan 2 kata (McCormick, 2020). SentencePiece memakai *underscore* untuk menandakan kata yang dipisah oleh spasi. Gambar 2.4 menunjukkan perbedaan hasil tokenisasi antara WordPiece dan SentencePiece pada kata "philosophy".



Gambar 2.4 Hasil tokenisasi kata "philosophy" (McCormick, 2020)

Pada Gambar 2.5 menunjukkan konfigurasi model ALBERT yang sudah diuji oleh Lan dkk (2019).

Model		Parameters	Layers	Hidden	Embedding	Parameter-sharing
BERT	base	108M	12	768	768	False
	large	334M	24	1024	1024	False
ALBERT	base	12M	12	768	128	True
	large	18M	24	1024	128	True
	xlarge	60M	24	2048	128	True
	xxlarge	235M	12	4096	128	True

Gambar 2.5 Konfigurasi ALBERT (Lan dkk., 2019)

Pada penelitian yang dilakukan oleh El-geish (2020) mengenai *Question Answering System* dengan menggunakan *stacking ensemble* dari dua model. Beberapa *fine-tuning* model yang digunakan untuk perbandingan pada penelitian tersebut ditunjukkan pada Gambar 2.6. Hasil dari penelitian menunjukkan model albert-xxlarge-v1 memiliki hasil evaluasi terbaik dan albert-base-v2 memiliki hasil evaluasi *Top-1 Answer* terbaik ketiga yang tidak berbeda jauh dengan albert-large-v2 walaupun memiliki parameter terkecil sesuai Gambar 2.5. Oleh karena itu, penelitian ini memutuskan untuk menggunakan model albert-xxlarge-v1 dan albert-base-v2.

	xlnet-base-cased	xlnet-large-cased	distilbert-base-uncased	roberta-base	albert-base-v2	albert-large-v2	albert-xxlarge-v1
Gradient Accumulation Steps	1	24	24	24	24	1	24
Learning Rate	3e-5	3e-5	3e-5	3e-5	3e-5	3e-5	3e-5
Max Answer Length	30	30	30	30	30	30	30
Max Query Length	64	64	64	64	64	64	64
Max Sequence Length	384	384	384	384	384	384	512
Training Epochs	4	3	4	4	3	5	4
Training Batch Size	16	4	32	16	8	8	1
Top-1 Dev EM Score	35.604	38.549	65.186	75.337	78.957	79.286	85.966
Top-1 Dev F1 Score	40.339	42.152	67.890	78.683	81.789	82.525	88.945
Top-8 Dev EM Score	—	—	92.991	94.373	94.538	93.205	94.965
Top-8 Dev F1 Score	—	—	94.245	95.372	95.447	94.230	95.473

Gambar 2.6 *Fine-tuning* Model dan Hasil Evaluasi (El-geish, 2020)

Penelitian ini juga memutuskan untuk memanfaatkan *fine-tuning* model pada penelitian yang dilakukan El-geish (2020). *Fine-tuning* model tersedia pada "https://huggingface.co/models?search=elgeish" dengan konfigurasi yang ditunjukkan pada Gambar 2.7 untuk model albert-xxlarge-v1 dan Gambar 2.8 untuk model albert-base-v2.


```
{
  "do_lower_case": true,
  "doc_stride": 128,
  "fp16": false,
  "fp16_opt_level": "O1",
  "gradient_accumulation_steps": 24,
  "learning_rate": 3e-05,
  "max_answer_length": 30,
  "max_grad_norm": 1,
  "max_query_length": 64,
  "max_seq_length": 512,
  "model_name_or_path": "albert-xxlarge-v1",
  "model_type": "albert",
  "num_train_epochs": 4,
  "per_gpu_train_batch_size": 1,
  "save_steps": 1000,
  "seed": 42,
  "train_batch_size": 1,
  "version_2_with_negative": true,
  "warmup_steps": 814,
  "weight_decay": 0
}
```

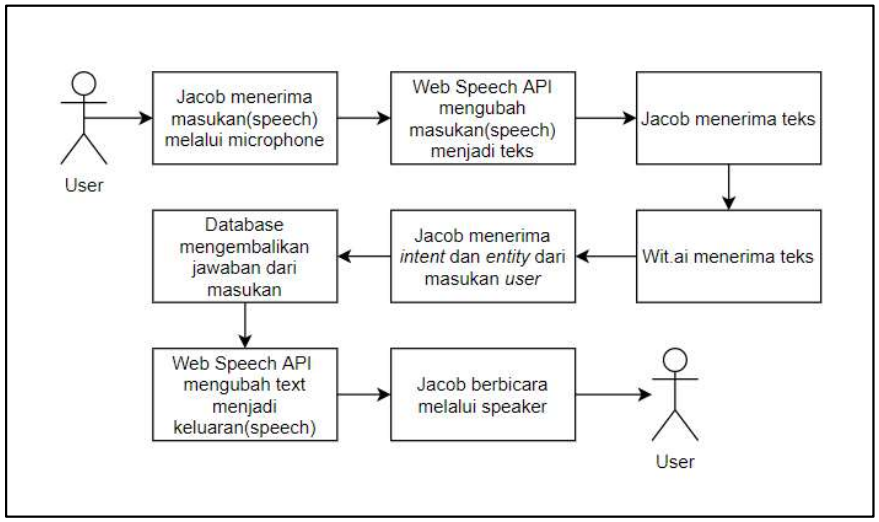

Gambar 2.7 Konfigurasi *Fine-tuning* albert-xxlarge-v1 (El-geish, 2020)


```
{
  "do_lower_case": true,
  "doc_stride": 128,
  "fp16": false,
  "fp16_opt_level": "O1",
  "gradient_accumulation_steps": 24,
  "learning_rate": 3e-05,
  "max_answer_length": 30,
  "max_grad_norm": 1,
  "max_query_length": 64,
  "max_seq_length": 384,
  "model_name_or_path": "albert-base-v2",
  "model_type": "albert",
  "num_train_epochs": 3,
  "per_gpu_train_batch_size": 8,
  "save_steps": 5000,
  "seed": 42,
  "train_batch_size": 8,
  "version_2_with_negative": true,
  "warmup_steps": 0,
  "weight_decay": 0
}
```


Gambar 2.8 Konfigurasi *Fine-tuning* albert-base-v2 (El-geish, 2020)

2.3 Jacob

Jacob adalah nama dari *voice chatbot* yang menyediakan informasi tentang program *Dual Degree* Informatika Universitas Multimedia Nusantara (Wijaya dan Wicaksana, 2019). Jacob tercipta karena Kepala Program Studi Informatika dan Kepala Departemen *Marketing* Universitas Multimedia Nusantara mengharapkan sebuah *voice chatbot* yang menggunakan Bahasa Inggris dapat memberikan informasi mengenai program *dual degree* Universitas Multimedia Nusantara. *Voice chatbot* yang dibuat diharapkan dapat melakukan pemasaran mengenai *joint-degree* Universitas Multimedia Nusantara secara internasional. Rancangan umum interaksi *user* dengan *voice chatbot* Jacob ada pada Gambar 2.9.



Gambar 2.9 Rancangan Umum Interaksi *User* dengan Jacob (Wijaya dan Wicaksana, 2019)

Jacob dikembangkan dalam basis web dengan menggunakan bahasa pemrograman PHP dengan *framework* Laravel dan Python dengan *framework* Flask (Wijaya dan Wicaksana, 2019). Jacob memiliki 4 modul penting antara lain:

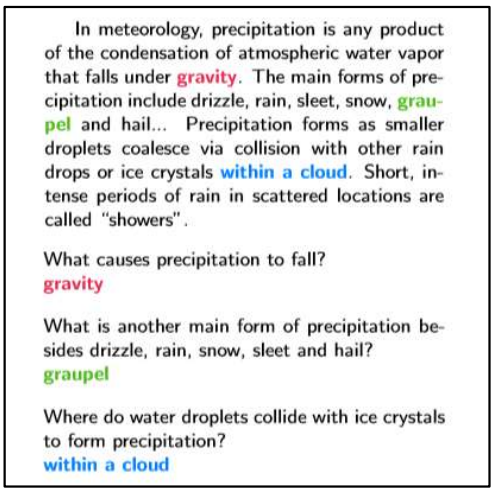
1. Modul Jacob, modul yang berfungsi sebagai *voice chatbot* pada dasarnya, yaitu menerjemahkan masukan dari *user* sehingga diperoleh inti dari masukan tersebut dan mengidentifikasi nama pemakai.
2. Modul Cleveree, modul yang berfungsi sebagai fitur "smart" dari *voice chatbot*, yaitu berupa *automatic summarization* dan parafrase jawaban.
3. Modul Vision, modul yang berfungsi sebagai fitur "penglihatan", yaitu berupa *face recognition*.
4. Modul Virtual Character, modul yang berfungsi untuk menampilkan animasi 3D dari karakter Jacob.

2.4 SQuAD

Stanford Question Answering Dataset (SQuAD) merupakan *reading comprehension dataset* yang terdiri lebih dari 100000 pertanyaan yang diajukan oleh *crowdworkers* pada artikel-artikel Wikipedia, dimana jawaban dari setiap pertanyaan ada didalam artikel tersebut (Rajpurkar dkk., 2016). SQuAD dibuat untuk memenuhi kebutuhan akan *dataset* dengan kualitas bagus dan besar. Contoh data dari SQuAD dapat dilihat pada Gambar 2.10. Tahapan pengumpulan *dataset* dilakukan dalam 3 tahap yaitu:

- *passage curation,* mengumpulkan 10000 artikel Wikipedia teratas untuk diambil paragraf yang dianggap penting,
- *question-answer collection,* meminta *crowdworkers* untuk memberikan pertanyaan dan menjawab 5 pertanyaan dari hasil yang didapat di *passage curation,* dan

- *additional answers collection*, meminta *crowdworkers* untuk memberikan setidaknya 2 jawaban tambahan untuk mendapatkan indikasi dari kinerja manusia.


In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under gravity. The main forms of precipitation include drizzle, rain, sleet, snow, graupel and hail... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals within a cloud. Short, intense periods of rain in scattered locations are called "showers".

What causes precipitation to fall?
gravity

What is another main form of precipitation besides drizzle, rain, snow, sleet and hail?
graupel

Where do water droplets collide with ice crystals to form precipitation?
within a cloud

Gambar 2.10 Contoh *question-answer pairs* dari SQuAD (Rajpurkar dkk., 2016)

Sekarang SQuAD telah dikembangkan menjadi 2 versi, yaitu versi 1.1 dan versi 2.0 (Rajpurkar dkk., 2018). SQuAD 2.0 dibuat untuk mengatasi kelemahan SQuAD 1.1 yang hanya focus pada pertanyaan yang bisa dijawab dan mengabaikan pertanyaan yang tidak bisa dijawab. SQuAD 2.0 terdiri dari gabungan SQuAD 1.1 dan tambahan 50000 pertanyaan yang tidak bisa dijawab. Tabel 2.1 menunjukkan perbandingan jumlah data pada *dataset* SQuAD 1.1 dengan SQuAD 2.0.

Tabel 2.1 Perbandingan jumlah data SQuAD 1.1 dengan SQuAD 2.0 (Rajpurkar dkk., 2018)

	SQuAD 1.1	SQuAD 2.0
Train		
Total examples	87.599	130.319
Negative examples	0	43.498
Total articles	442	442
Articles with negatives	0	285

Tabel 2.1 Perbandingan jumlah data SQuAD 1.1 dengan SQuAD 2.0 (lanjutan)

Development		
Total examples	10.570	11.873
Negative examples	0	5.945
Total articles	48	35
Articles with negatives	0	35
Test		
Total examples	9.533	8.862
Negative examples	0	4.332
Total articles	46	28
Articles with negatives	0	28

2.5 Akurasi, F-Score, & User Time

Akurasi dan *F-Score* merupakan cara mengukur evaluasi performa yang cukup umum (Tharwat, 2018). Akurasi merupakan rasio prediksi yang benar di keseluruhan data.

$$accuracy = \frac{tp + tn}{tp + fp + fn + tn} \quad \ldots(2.1)$$

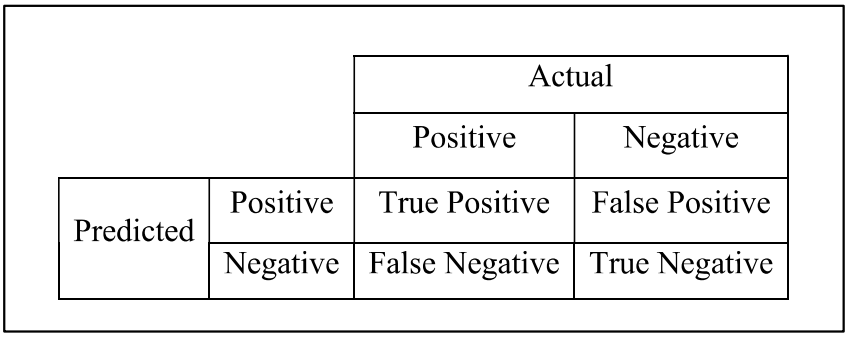
Sedangkan *F-Score* merupakan perbandingan rata-rata *precision* dan *recall* yang lebih berguna dibandingkan dengan akurasi ketika terdapat distribusi *class* yang tidak seimbang (Alexander dkk., 2019).

$$precision = \frac{tp}{tp + fp} \quad \ldots(2.2)$$

$$recall = \frac{tp}{tp + fn} \quad \ldots(2.3)$$

$$F - score = \frac{2 \times precision \times recall}{precision + recall} \quad \ldots(2.4)$$

Dimana tp adalah *True Positive,* tn adalah *True Negative,* fp adalah *False Positive,* dan fn adalah *False Negative* yang didapat dari *confusion matrix* pada Gambar 2.11.



Gambar 2.11 *Confusion Matrix* (Alexander dkk., 2019)

User time adalah *wall-clock time* yang ada di dalam *web service* (Wicaksana dan Tang, 2017). *Wall-clock time* yang dimaksud adalah rentang waktu antara sebuah *task* mulai hingga *task* tersebut selesai (Roussel, 2011). Pencatatan *user time* digunakan untuk membantu pengamatan performa dari *web service*. Pada bahasa pemrograman Python *user time* dapat dihitung menggunakan *time.perf_counter*() (Poole dan Mackworth, 2020).