## Overall task

## Eksplorasi data / data understanding

## Preprocessing data

## Model CNN

## Model RNN

## Model BERT

## Model SVM

## Feature extraction

## Feature embedding

## Performance Metrics

## Poster

## 

## Persiapan data

Berbagai teknik penyiapan data, khususnya untuk teks bahasa Indonesia:

### **Data Understanding:** lebih ke pemahaman data biar ngerti data ini harus di preprocess kaya gimana

1. Statistical visualisation
   1. Word frequency,
   2. sentence length,
   3. average word length,
   4. listing and counting stopwords
   5. Can use word cloud
2. Topic modelling (opsional)
   1. Ini malah udah modelnya
   2. Unsupervised learning technique to extract main topic
   3. Can use LDA (gensim.models.LdaMulticore)
3. Named Entity Recognition (kalo bisa bagus, tapi opsional)
   1. To find out what or who is this text talking about
   2. Or count the type to know if the text is talking more about a person, a place, or anything else
   3. Can visualize most common tokens per entity, e.g. searching which places appear the most among news headlines
   4. Can use Spacy

### 

### **Preprocessing: bikin fungsi**”nya dulu aja biar ntar tinggal pake

1. Normalisasi
   1. Normalisasi adalah mengubah token menjadi bentuk dasarnya. Dalam normalisasi, bentuk infleksi suatu kata dihilangkan sehingga diperoleh bentuk dasarnya. Contohnya antinasionalis menjadi nasional
   2. Kalimat diubah seluruhnya menjadi lowercase
   3. Seluruh unsur kalimat yang bukan huruf dihilangkan (seperti emoticon, simbol).
   4. Mengubah kata-kata kalimat menjadi kata baku penggantinya.
2. Tokenization
   1. Memisahkan menjadi beberapa kata, bisa unigram, bigram, trigram, ngram
   2. Mungkin memisahkan kata yang harusnya tidak dipisah (e.g. New York)
   3. Might remove punctuation, misal *dr.* harusnya titiknya ga dihapus
3. Stop words removal: bikin algoritma, list belakangan
   1. Menghilangkan “and”, “the”, “to” kalo di bahasa Inggris; so removing widespread and frequent terms that are not informative
   2. Dilakukan dengan look up ke list stopword yang didefinisikan sebelumnya
   3. Isi list perlu diperhatikan dengan hati-hati, bisa jadi ada kata yang dianggap stopword tapi ternayta penting. Perlu diadjust ke tiap teks
4. Stemming: sastrawi
   1. Slicing the end or the beginning of words to remove affixes. Untuk mengurangi jumlah kata-kata yang harus diproses (e.g. treat *writing* and *write* as the same word = jadi *writ*)
   2. Might change the word and sentence's meaning. e.g. *caring* distem malah jadi *car*, *news* distem malah jadi *new*. Tangani dengan menambahkan atau mengurangi *affix rules*
5. Lemmatization
   1. Slicing affix too, tapi kalo ini sambil look up ke kamus buat cari kata dasar. Misalnya *went* jadi *go, best* jadi *good.*
   2. Karena take context, it can discriminate identical words that have different meaning based on context
   3. Demands more computational power than stemming algorithm

### Feature Extraction (dua-duanya harus)

<https://medium.com/@eiki1212/feature-extraction-in-natural-language-processing-with-python-59c7cdcaf064>

Feature Extraction memiliki dua metode utama: bag-of-words, dan word embedding. Keduanya umum digunakan dan memiliki pendekatan yang berbeda.

1. Bag-of-Words ***bisa lanjut ke tf-idf, word frequency.*** *perlu juga karena word embedding agak lama, kalo gacukup bikin embedding bisa turun ke bag of words*

Bag of Words adalah model representasi data, yang hanya menghitung berapa kali sebuah kata muncul dalam sebuah dokumen. Bag-of-Words umumnya digunakan dalam pengelompokan, klasifikasi, dan pemodelan topik dengan menimbang kata-kata khusus dan terminologi yang relevan. Flow:

#Sample Sentence

John likes to watch movies. Mary likes movies too.

(1)Tokenizing. "John","likes","to","watch","movies","Mary","likes","movies","too"

(2) Creating word set(dictionary).

"John","likes","to","watch","Mary","likes","movies","too"

(3) Counting occurrences and transform to Bag-of-words.

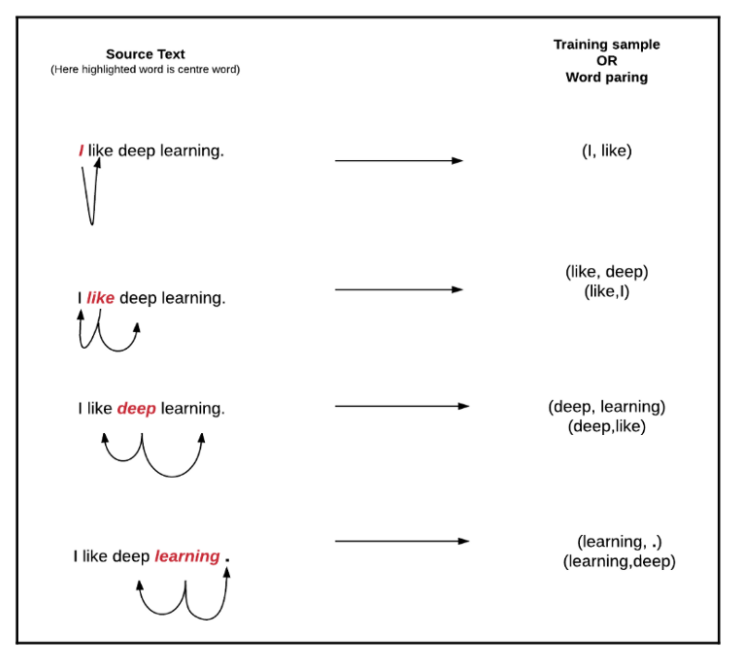
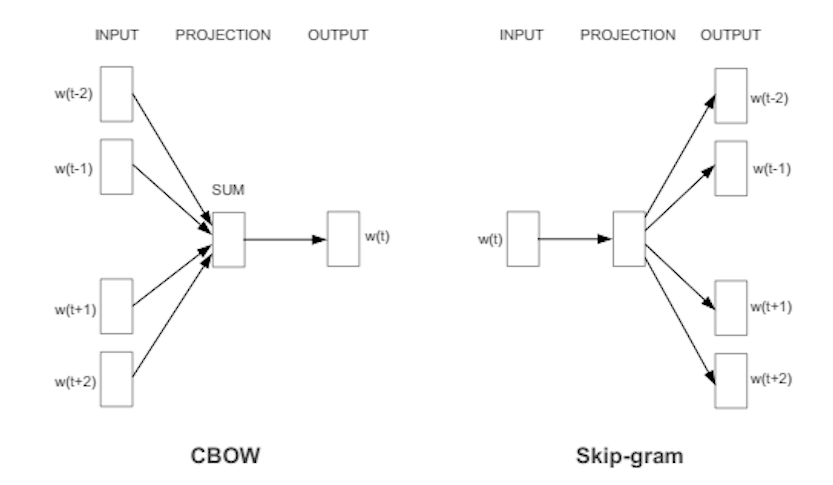
{"John":1,"likes":2,"to":1,"watch":1,"movies":2,"Mary":1,"too":1}

1. Word embedding

<https://medium.com/@vishwasbhanawat/the-architecture-of-word2vec-78659ceb6638>

Word embedding merupakan salah satu representasi data dalam model ruang vektor. Sebelumnya, bag-of-Words hanya mewakili jumlah kemunculan kata tanpa hubungan dan konteks. Di sisi lain, penyematan Word mempertahankan konteks dan hubungan kata-kata sehingga mendeteksi kata-kata serupa secara lebih akurat.

Word2vec adalah salah satu implementasi paling populer dari penyisipan kata. Ini menjelaskan penyematan kata dengan jaringan neural network dua lapis untuk mengenali makna konteks. Word2vec pandai mengelompokkan kata-kata yang mirip dan membuat tebakan yang sangat akurat tentang arti kata berdasarkan konteks



### Feature Embedding

<https://machinelearningmastery.com/what-are-word-embeddings/>

Representation for text; words with similar meaning have similar representation. Individual words represented as real-valued vector

Algorithms:

1. Embedding layer
   1. Learned jointly with a neural network model on the front end
   2. Require documents to be cleaned and prepared, contoh tiap kata di one-hot encoded.
   3. Vector will be initialized with small random numbers, size of vector space termasuk parameter
   4. One-hot encoded words will be mapped to word vectors.
   5. Require a lot of training data, can be slow; but will learn an embedding for the specific text and task
2. Word2Vec (<https://github.com/deryrahman/word2vec-bahasa-indonesia>) (recommend ini)
   1. Standalone word embedding from a text corpus, a response to make neural-network based training more efficient
   2. Has two different model:
      1. continuous bag of words: predict current word based on context
      2. continuous skipgram: predict surrounding word given current word
   3. Can be learned efficiently (low space & time complexity), allowing more dimensions from much larger text
3. GloVe
   1. Extension to word2vec, developed using matrix factorization
   2. Constructs an explicit word-context matrix using statistic across whole text corpus

How to use:

1. Learn / construct
   1. learn as a standalone, then saved to be used as a part of the model later
   2. learn jointly as a part of large task specific model. Good if only want to use the embedding on one task
2. Reuse embedding
   1. Use available pre-trained word embedding (word2vec and glove)
   2. Two options:
      1. Static: used as a component, pick this if the embedding is a good suit for the problem and give good result
      2. Updated: used to seed the model, but updated jointly during model training. Good if we want to get the most out of the model
3. Better to start with fast option, like pre-trained embedding, then adjust or use a new one of the pre-trained one doesn’t fit

## Membangun model

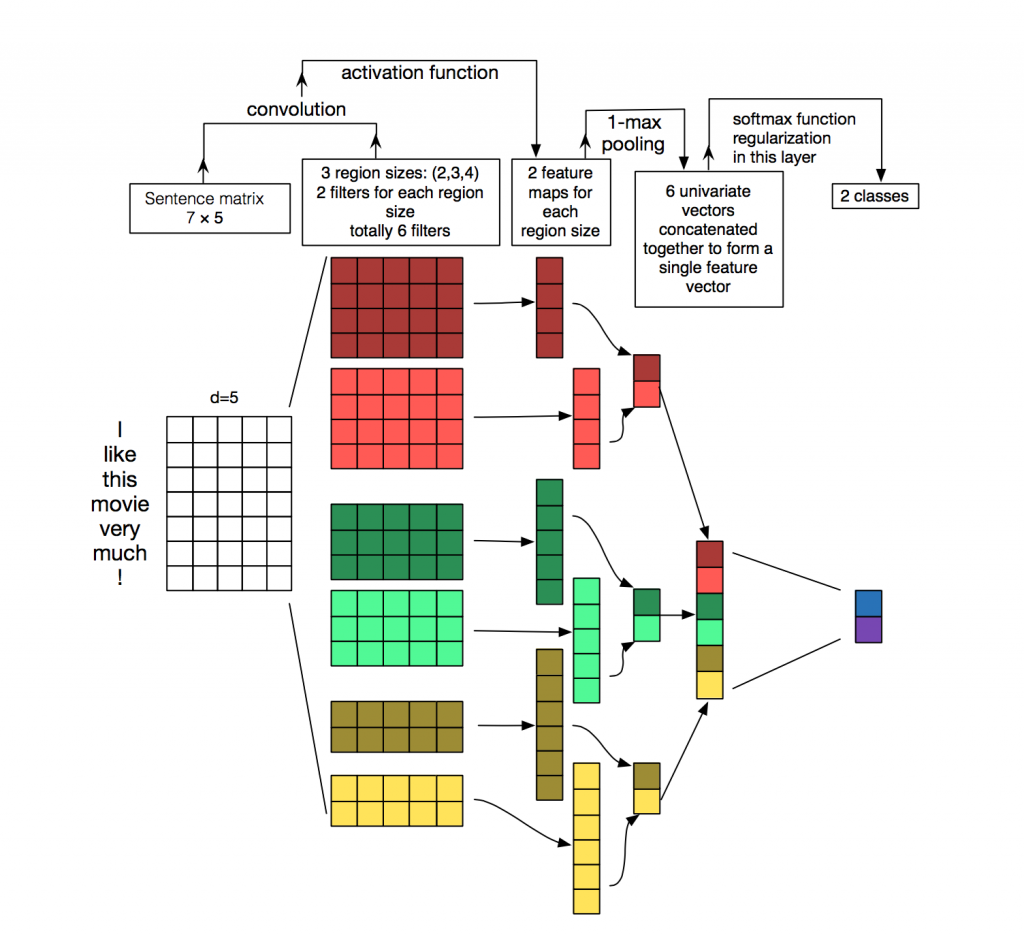
Berbagai teknik konstruksi model + tuning parameternya

### 

### CNN

CNN pada dasarnya hanyalah beberapa lapisan konvolusi dengan fungsi aktivasi nonlinier seperti ReLU atau tanh yang diterapkan pada hasil. Di CNN, diggunakan konvolusi di atas lapisan input untuk menghitung output. Ini menghasilkan koneksi lokal, di mana setiap wilayah input terhubung ke neuron di output. Setiap lapisan menerapkan filter yang berbeda, dan menggabungkan hasilnya.

Input untuk sebagian besar tugas NLP adalah kalimat yang direpresentasikan sebagai matriks. Setiap baris matriks sesuai dengan satu token, dapat berupa kata, atau karakter. Artinya, setiap baris adalah vektor yang mewakili sebuah kata. Biasanya, vektor-vektor ini adalah penyisipan kata (representasi dimensi rendah) seperti word2vec atau GloVe, tetapi mereka juga bisa menjadi vektor one-hot yang mengindeks kata ke dalam kosa kata. Untuk kalimat 10 kata menggunakan embedding 100 dimensi, kita akan memiliki matriks 10x100 sebagai input kita.



### RNN

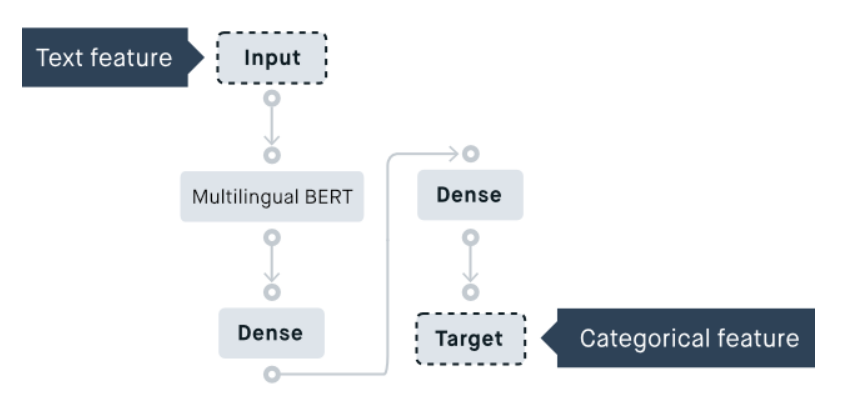
* Ideal for problems where sequence is more important
* Among text usage, best for sequence labelling, text classification, text generation

### BERT

<https://github.com/nxs5899/Named-Entity-Recognition_DeepLearning-keras>

<https://github.com/prateekjoshi565/Fine-Tuning-BERT/blob/master/Fine_Tuning_BERT_for_Spam_Classification.ipynb>

<https://towardsdatascience.com/bert-text-classification-using-pytorch-723dfb8b6b5b>



### SVM

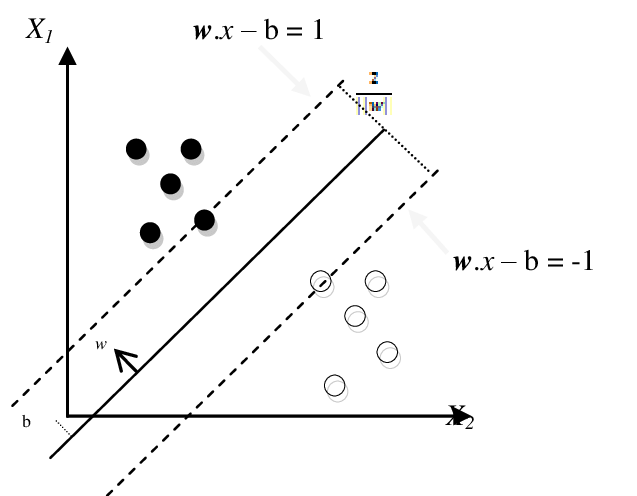
Contoh percobaan klasifikasi emosi twitter dengan svm:

<https://colab.research.google.com/drive/13iMEFoNWbC8lEE314lXy3-64qPIzI4R2?usp=sharing>

<https://medium.com/@bedigunjit/simple-guide-to-text-classification-nlp-using-svm-and-naive-bayes-with-python-421db3a72d34>

<https://medium.com/@vasista/sentiment-analysis-using-svm-338d418e3ff1>

SVM adalah model pembelajaran dengan algoritma pembelajaran terkait yang menganalisis data untuk analisis klasifikasi dan regresi.



## Evaluasi

Pemahaman dan mengukur berbagai performance metrics

1. Accuracy
   1. Ratio of correctly predicted : total observations
   2. Best for even data
2. Precision
   1. Ratio of correctly predicted positives : total predicted observations
   2. Out of all passengers labelled survived, how many actually survived?
   3. High precision = low false positive
3. Recall (sensitivity)
   1. Ratio of correctly predicted positives : total positives
   2. Out of all passengers that survived, how many did we label?
4. F1 Score
   1. Weighted average of precision and recall. Takes FP and FN
   2. More useful even if we have uneven class distribution
5. Area Under Curve (AUC)
   1. For binary classification problem
   2. Plots TP rate against FP rate, measure ability to distinguish classes
   3. AUC 1 = perfectly distinguish (all positives are positive dan sebaliknya)  
      AUC 0 = all positives are negative dan sebaliknya  
      AUC 0.5 = classifier predicts random class or constant class
6. Mean Reciprocal Rank (MRR)
   1. Return ranked list of answers to queries
   2. Typically used in information retrieval tasks
   3. If no correct answer, reciprocal rank is 0
7. Root Mean Square Error
   1. When the predicted outcome is a real value (bilangan pecahan)
   2. Used in regression problems to check error on predicting next value

## Referensi

* [Your Guide to Natural Language Processing (NLP) | by Diego Lopez Yse](https://towardsdatascience.com/your-guide-to-natural-language-processing-nlp-48ea2511f6e1)
* [Exploratory Data Analysis for Natural Language Processing: A Complete Guide to Python Tools - neptune.ai](https://neptune.ai/blog/exploratory-data-analysis-natural-language-processing-tools)
* [Accuracy, Precision, Recall & F1 Score: Interpretation of Performance Measures - Exsilio Blog](https://blog.exsilio.com/all/accuracy-precision-recall-f1-score-interpretation-of-performance-measures/)
* [Recurrent Neural Networks and Natural Language Processing. | by Christopher Thomas BSc Hons. MIAP](https://towardsdatascience.com/recurrent-neural-networks-and-natural-language-processing-73af640c2aa1)